```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_stata('wage.dta')
```

```
df.head()
```

	w	fe	nw	un	ed	ex	age	wk
0	11.55	1.0	0.0	0.0	12.0	20.0	38.0	1.0
1	5.00	0.0	0.0	0.0	9.0	9.0	24.0	0.0
2	12.00	0.0	0.0	0.0	16.0	15.0	37.0	1.0
3	7.00	0.0	1.0	1.0	14.0	38.0	58.0	0.0
4	21.15	1.0	1.0	0.0	16.0	19.0	41.0	1.0

```
df.describe()
```

	w	fe	nw	un	ed	ex	age	wk
count	1289.000000	1289.000000	1289.000000	1289.000000	1289.000000	1289.000000	1289.000000	1289.000000
mean	12.365849	0.497285	0.152832	0.159038	13.145074	18.789759	37.934834	0.407292
std	7.896350	0.500187	0.359965	0.365853	2.813823	11.662837	11.494278	0.491521
min	0.840000	0.000000	0.000000	0.000000	0.000000	0.000000	18.000000	0.000000
25%	6.920000	0.000000	0.000000	0.000000	12.000000	9.000000	29.000000	0.000000
50%	10.080000	0.000000	0.000000	0.000000	12.000000	18.000000	37.000000	0.000000
75%	15.630000	1.000000	0.000000	0.000000	16.000000	27.000000	47.000000	1.000000
max	64.080002	1.000000	1.000000	1.000000	20.000000	56.000000	65.000000	1.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1289 entries, 0 to 1288
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   w       1289 non-null   float32
 1   fe      1289 non-null   float32
 2   nw      1289 non-null   float32
 3   un      1289 non-null   float32
 4   ed      1289 non-null   float32
 5   ex      1289 non-null   float32
 6   age     1289 non-null   float32
 7   wk      1289 non-null   float32
dtypes: float32(8)
memory usage: 40.4 KB
```

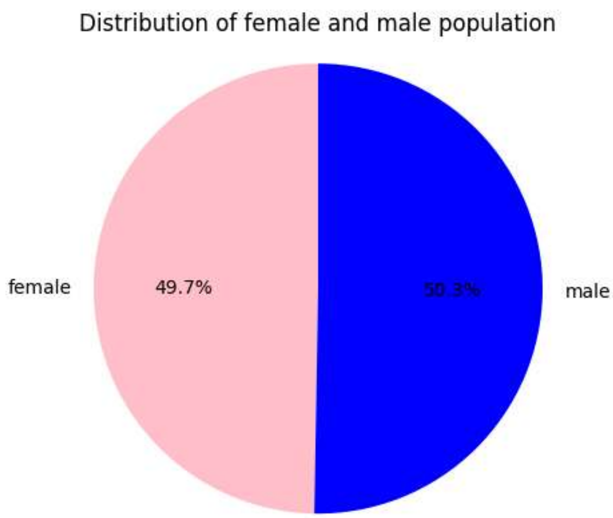
```
df.to_csv('wage.csv', index= False)
```

```
df['ln(wage)'] = np.log(df['w'])
```

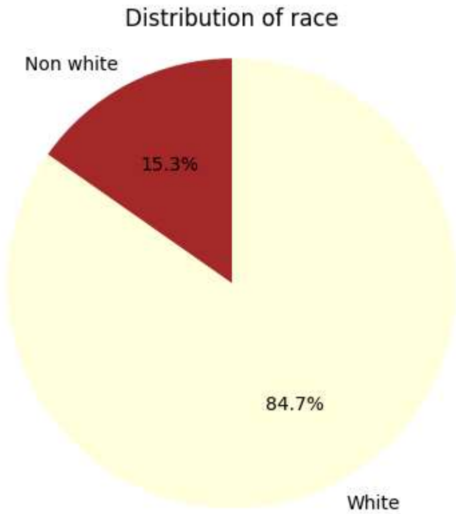
```
df.head()
```

	w	fe	nw	un	ed	ex	age	wk	ln(wage)
0	11.55	1.0	0.0	0.0	12.0	20.0	38.0	1.0	2.446686
1	5.00	0.0	0.0	0.0	9.0	9.0	24.0	0.0	1.609438
2	12.00	0.0	0.0	0.0	16.0	15.0	37.0	1.0	2.484907
3	7.00	0.0	1.0	1.0	14.0	38.0	58.0	0.0	1.945910
4	21.15	1.0	1.0	0.0	16.0	19.0	41.0	1.0	3.051640

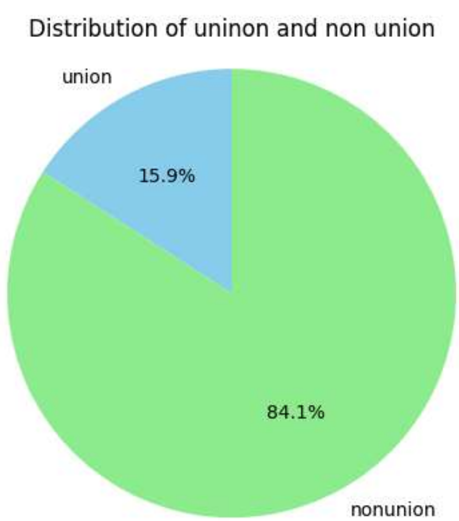
```
# Count the occurrences of female values
fe_counts = df['fe'].value_counts()

# Define the labels and sizes for the pie chart
labels = ['female', 'male']
sizes = [fe_counts[1.0], fe_counts[0.0]]

# Plot the pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['pink', 'blue'])

# Equal aspect ratio ensures that pie chart is drawn as a circle.
plt.axis('equal')

# Show the plot
plt.title('Distribution of female and male population')
plt.show()
```




```
# Count the occurrences of fe values
nw_counts = df['nw'].value_counts()

# Define the labels and sizes for the pie chart
labels = ['Non white', 'White']
sizes = [nw_counts[1.0], nw_counts[0.0]]

# Plot the pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['brown', 'lightyellow'])

# Equal aspect ratio ensures that pie chart is drawn as a circle.
plt.axis('equal')

# Show the plot
plt.title('Distribution of race')
plt.show()
```

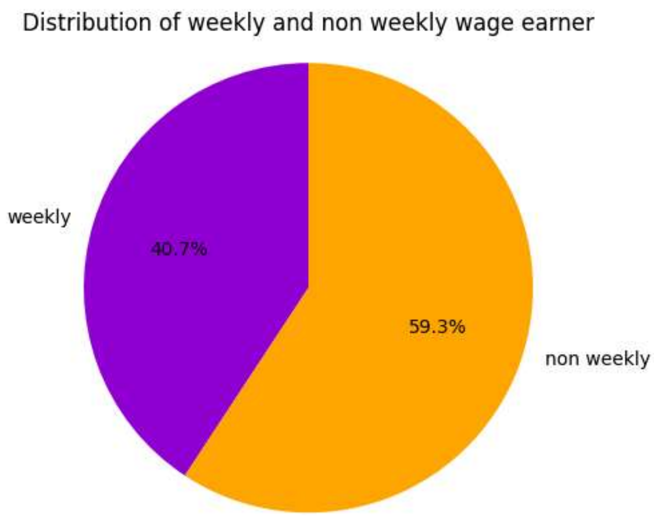



```
 #Count the occurrences of un values
un_counts = df['un'].value_counts()

# Define the labels and sizes for the pie chart
labels = ['union', 'nonunion']
sizes = [un_counts[1.0], un_counts[0.0]]

# Plot the pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightgreen'])

# Equal aspect ratio ensures that pie chart is drawn as a circle.
plt.axis('equal')

# Show the plot
plt.title('Distribution of uninon and non union')
plt.show()
```

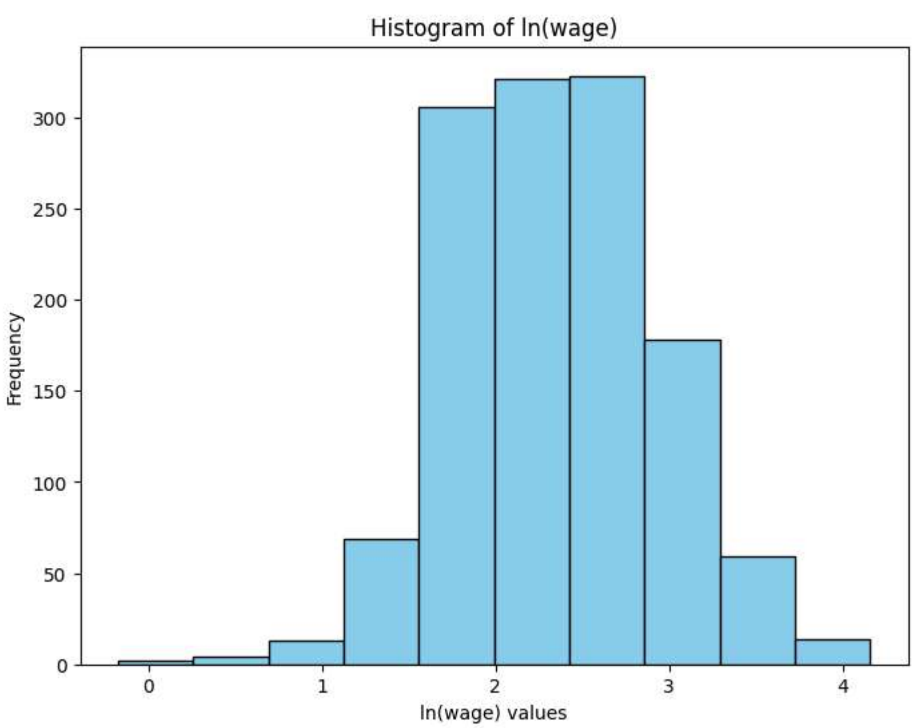



```
# Count the occurrences of fe values
wk_counts = df['wk'].value_counts()

# Define the labels and sizes for the pie chart
labels = ['weekly', 'non weekly']
sizes = [wk_counts[1.0], wk_counts[0.0]]

# Plot the pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['darkviolet', 'orange'])

# Equal aspect ratio ensures that pie chart is drawn as a circle.
plt.axis('equal')
```

```
# Show the plot
plt.title('Distribution of weekly and non weekly wage earner')
plt.show()
```




```
# Plot the histogram for the 'ln(wage)' column
plt.figure(figsize=(8, 6))  # Optional: Adjust figure size
plt.hist(df['ln(wage)'], bins=10, color='skyblue', edgecolor='black')

# Add titles and labels
plt.title('Histogram of ln(wage)')
plt.xlabel('ln(wage) values')
plt.ylabel('Frequency')

# Show the plot
plt.show()
```




```
Q1 = df['ln(wage)'].quantile(0.25)
Q3 = df['ln(wage)'].quantile(0.75)
IQR = Q3 - Q1

# Define outlier thresholds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
# Identify outliers
outliers = df[(df['ln(wage)'] < lower_bound) | (df['ln(wage)'] > upper_bound)]
print(outliers)
```

```
              w   fe   nw   un    ed    ex   age   wk  ln(wage)
97     2.000000  1.0  0.0  0.0   5.0  14.0  25.0  1.0  0.693147
372    0.840000  1.0  0.0  0.0  10.0  35.0  51.0  1.0 -0.174353
715   64.080002  1.0  0.0  0.0  16.0  12.0  34.0  1.0  4.160132
722    1.570000  0.0  1.0  0.0  16.0   6.0  28.0  1.0  0.451076
904    1.500000  1.0  0.0  0.0  14.0  17.0  37.0  0.0  0.405465
933    2.000000  0.0  0.0  0.0  11.0   1.0  18.0  0.0  0.693147
956    1.150000  0.0  0.0  0.0  12.0  43.0  61.0  1.0  0.139762
1256   1.670000  0.0  0.0  0.0  16.0  10.0  32.0  1.0  0.512824
1284   1.670000  1.0  0.0  0.0  10.0  45.0  61.0  1.0  0.512824
```

```
# Remove outliers
df_cleaned = df[(df['ln(wage)'] >= lower_bound) & (df['ln(wage)'] <= upper_bound)]

# Display the cleaned DataFrame
print(df_cleaned)
```

```
          w   fe   nw   un    ed    ex   age   wk  ln(wage)
0     11.55  1.0  0.0  0.0  12.0  20.0  38.0  1.0  2.446686
1      5.00  0.0  0.0  0.0   9.0   9.0  24.0  0.0  1.609438
2     12.00  0.0  0.0  0.0  16.0  15.0  37.0  1.0  2.484907
3      7.00  0.0  1.0  1.0  14.0  38.0  58.0  0.0  1.945910
4     21.15  1.0  1.0  0.0  16.0  19.0  41.0  1.0  3.051640
...     ...  ...  ...  ...   ...   ...   ...  ...       ...
1283  13.14  1.0  0.0  0.0  12.0  30.0  48.0  0.0  2.575661
1285   7.00  0.0  0.0  0.0  12.0   7.0  25.0  0.0  1.945910
1286  14.90  0.0  0.0  0.0  16.0   9.0  31.0  1.0  2.701361
1287   5.90  1.0  0.0  0.0  12.0  31.0  49.0  0.0  1.774952
1288  10.00  0.0  0.0  1.0  12.0  20.0  38.0  1.0  2.302585

[1280 rows x 9 columns]
```

```
df1 = df_cleaned
```

```
df1.columns
```

```
Index(['w', 'fe', 'nw', 'un', 'ed', 'ex', 'age', 'wk', 'ln(wage)'], dtype='object')
```

```
X1=df1[['fe', 'nw', 'un', 'ed', 'ex', 'age', 'wk']]
```

```
X1.drop('age', axis=1, inplace=True)
```

```
<ipython-input-126-ff48928db454>:1: 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-cc
  X1.drop('age', axis=1, inplace=True)
```

```
Y=df1['ln(wage)']
```

```
X1.head()
```

	fe	nw	un	ed	ex	wk
0	1.0	0.0	0.0	12.0	20.0	1.0
1	0.0	0.0	0.0	9.0	9.0	0.0
2	0.0	0.0	0.0	16.0	15.0	1.0
3	0.0	1.0	1.0	14.0	38.0	0.0
4	1.0	1.0	0.0	16.0	19.0	1.0

```
import statsmodels.api as sm
```

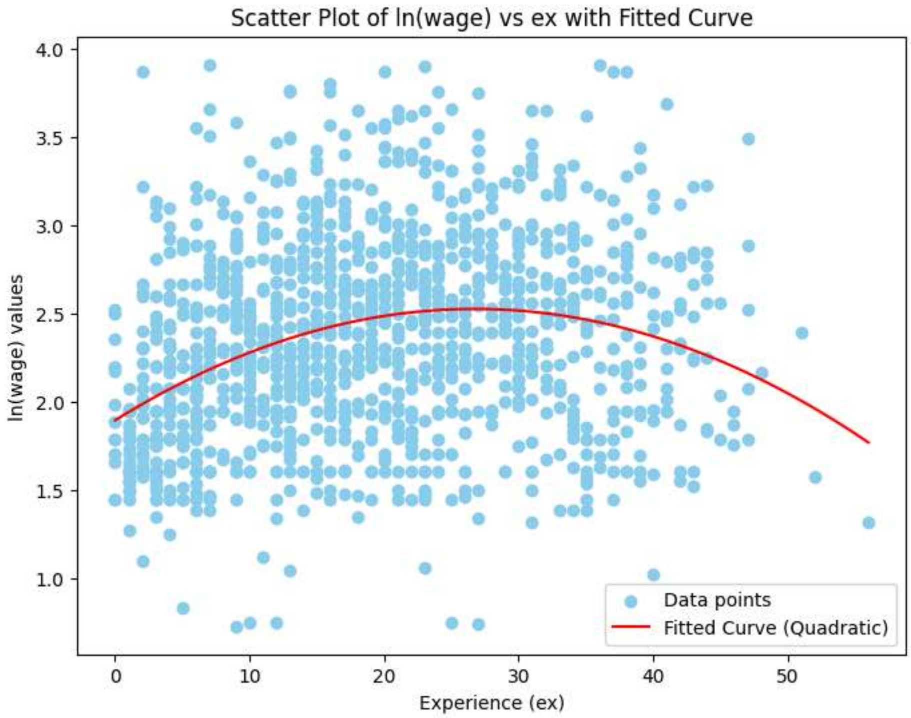
```
reg_result1 = sm.OLS(Y, sm.add_constant(X1)).fit()
```

```
print(reg_result1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln(wage)   R-squared:                       0.408
Model:                            OLS   Adj. R-squared:                  0.405
Method:                 Least Squares   F-statistic:                     146.0
Date:                Tue, 24 Sep 2024   Prob (F-statistic):          5.60e-141
Time:                        07:00:18   Log-Likelihood:                -750.43
No. Observations:                1280   AIC:                             1515.
Df Residuals:                    1273   BIC:                             1551.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0743      0.070     15.281      0.000       0.936       1.212
fe            -0.2390      0.025     -9.721      0.000      -0.287      -0.191
nw            -0.1222      0.034     -3.568      0.000      -0.189      -0.055
un             0.1941      0.034      5.694      0.000       0.127       0.261
ed             0.0812      0.005     16.779      0.000       0.072       0.091
ex             0.0115      0.001     10.491      0.000       0.009       0.014
wk             0.2497      0.027      9.124      0.000       0.196       0.303
==============================================================================
Omnibus:                       24.780   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               41.698
Skew:                          -0.138   Prob(JB):                     8.82e-10
Kurtosis:                       3.840   Cond. No.                         146.
==============================================================================

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

Start coding or generate with AI.

```
# Scatter plot of 'ln(wage)' vs 'ex'
plt.figure(figsize=(8, 6))
plt.scatter(df1['ex'], df1['ln(wage)'], color='skyblue', label='Data points')

# Fit a second-degree polynomial (quadratic curve)
coefficients = np.polyfit(df1['ex'], df1['ln(wage)'], 2)
polynomial = np.poly1d(coefficients)

# Generate x values (ex range) for plotting the fitted curve
x_vals = np.linspace(df1['ex'].min(), df1['ex'].max(), 100)
y_vals = polynomial(x_vals)

# Plot the fitted curve
plt.plot(x_vals, y_vals, color='red', label='Fitted Curve (Quadratic)')

# Add titles and labels
plt.title('Scatter Plot of ln(wage) vs ex with Fitted Curve')
plt.xlabel('Experience (ex)')
plt.ylabel('ln(wage) values')
plt.legend()

# Show the plot
plt.show()
```




```
X2=X1
```

```
X2['ex^2'] = X2['ex']**2
```

```
<ipython-input-134-415623a447f6>: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-cc
  X2['ex^2'] = X2['ex']**2
```

```
X2.head()
```

	fe	nw	un	ed	ex	wk	ex^2
0	1.0	0.0	0.0	12.0	20.0	1.0	400.0
1	0.0	0.0	0.0	9.0	9.0	0.0	81.0
2	0.0	0.0	0.0	16.0	15.0	1.0	225.0
3	0.0	1.0	1.0	14.0	38.0	0.0	1444.0
4	1.0	1.0	0.0	16.0	19.0	1.0	361.0

```
reg_result2 = sm.OLS(Y, sm.add_constant(X2)).fit()
```

```
print(reg_result2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln(wage)   R-squared:                       0.427
Model:                            OLS   Adj. R-squared:                  0.424
Method:                 Least Squares   F-statistic:                     135.6
Date:                Tue, 24 Sep 2024   Prob (F-statistic):          3.77e-149
Time:                        07:00:19   Log-Likelihood:                -728.83
No. Observations:                1280   AIC:                             1474.
Df Residuals:                    1272   BIC:                             1515.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.9621      0.071     13.510      0.000       0.822       1.102
fe            -0.2331      0.024     -9.633      0.000      -0.281      -0.186
nw            -0.1199      0.034     -3.560      0.000      -0.186      -0.054
```

```
un             0.1880      0.034      5.606      0.000       0.122       0.254
ed             0.0775      0.005     16.184      0.000       0.068       0.087
ex             0.0342      0.004      9.506      0.000       0.027       0.041
wk             0.2435      0.027      9.040      0.000       0.191       0.296
ex^2          -0.0005   8.23e-05     -6.608      0.000      -0.001      -0.000
==============================================================================
Omnibus:                       34.727   Durbin-Watson:                   1.978
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               68.010
Skew:                          -0.156   Prob(JB):                     1.71e-15
Kurtosis:                       4.085   Cond. No.                     4.24e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
X3 = X2
```

```
X3['fe:nw'] = X3['fe']*X3['nw']
X3['fe:un'] = X2['fe']*X1['un']
X3['fe:ed'] = X2['fe']*X1['ed']
X3['fe:ex'] = X3['fe']*X1['ex']
X3['fe:wk'] = X3['fe']*X1['wk']
```

```
<ipython-input-139-fb238cc67dde>: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-cc
  X3['fe:nw'] = X3['fe']*X3['nw']
```

```
X1.head()
```

	fe	nw	un	ed	ex	wk	ex^2	fe:nw	fe:un	fe:ed	fe:ex	fe:wk
0	1.0	0.0	0.0	12.0	20.0	1.0	400.0	0.0	0.0	12.0	20.0	1.0
1	0.0	0.0	0.0	9.0	9.0	0.0	81.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	16.0	15.0	1.0	225.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.0	1.0	14.0	38.0	0.0	1444.0	0.0	0.0	0.0	0.0	0.0
4	1.0	1.0	0.0	16.0	19.0	1.0	361.0	1.0	0.0	16.0	19.0	1.0

```
reg_result3 = sm.OLS(Y, sm.add_constant(X3)).fit()
```

```
print(reg_result3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                ln(wage)   R-squared:                       0.430
Model:                             OLS   Adj. R-squared:                  0.425
Method:                  Least Squares   F-statistic:                     79.65
Date:                 Tue, 24 Sep 2024   Prob (F-statistic):          2.89e-145
Time:                         07:00:19   Log-Likelihood:                -725.89
No. Observations:                 1280   AIC:                             1478.
Df Residuals:                     1267   BIC:                             1545.
Df Model:                           12
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0023      0.095     10.508      0.000       0.815       1.189
fe            -0.3178      0.137     -2.322      0.020      -0.586      -0.049
nw            -0.1418      0.050     -2.835      0.005      -0.240      -0.044
un             0.1775      0.045      3.986      0.000       0.090       0.265
ed             0.0717      0.007     10.769      0.000       0.059       0.085
ex             0.0360      0.004      9.406      0.000       0.028       0.044
wk             0.2602      0.039      6.626      0.000       0.183       0.337
ex^2          -0.0005   8.27e-05     -6.568      0.000      -0.001      -0.000
fe:nw          0.0460      0.068      0.677      0.498      -0.087       0.179
fe:un          0.0171      0.068      0.251      0.802      -0.117       0.151
fe:ed          0.0116      0.010      1.206      0.228      -0.007       0.030
fe:ex         -0.0035      0.002     -1.609      0.108      -0.008       0.001
fe:wk         -0.0285      0.054     -0.527      0.598      -0.135       0.078
==============================================================================
```

```
Omnibus:                       35.547   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.193
Skew:                          -0.159   Prob(JB):                     5.73e-16
Kurtosis:                       4.102   Cond. No.                     9.24e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
X2
```

	fe	nw	un	ed	ex	wk	ex^2	fe:nw	fe:un	fe:ed	fe:ex	fe:wk
0	1.0	0.0	0.0	12.0	20.0	1.0	400.0	0.0	0.0	12.0	20.0	1.0
1	0.0	0.0	0.0	9.0	9.0	0.0	81.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	16.0	15.0	1.0	225.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.0	1.0	14.0	38.0	0.0	1444.0	0.0	0.0	0.0	0.0	0.0
4	1.0	1.0	0.0	16.0	19.0	1.0	361.0	1.0	0.0	16.0	19.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
1283	1.0	0.0	0.0	12.0	30.0	0.0	900.0	0.0	0.0	12.0	30.0	0.0
1285	0.0	0.0	0.0	12.0	7.0	0.0	49.0	0.0	0.0	0.0	0.0	0.0
1286	0.0	0.0	0.0	16.0	9.0	1.0	81.0	0.0	0.0	0.0	0.0	0.0
1287	1.0	0.0	0.0	12.0	31.0	0.0	961.0	0.0	0.0	12.0	31.0	0.0
1288	0.0	0.0	1.0	12.0	20.0	1.0	400.0	0.0	0.0	0.0	0.0	0.0

1280 rows × 12 columns

```
X2.columns
```

```
Index(['fe', 'nw', 'un', 'ed', 'ex', 'wk', 'ex^2', 'fe:nw', 'fe:un', 'fe:ed',
       'fe:ex', 'fe:wk'],
      dtype='object')
```

```
X4 = X2[['fe', 'nw', 'un', 'ed', 'ex', 'wk', 'ex^2']]
```

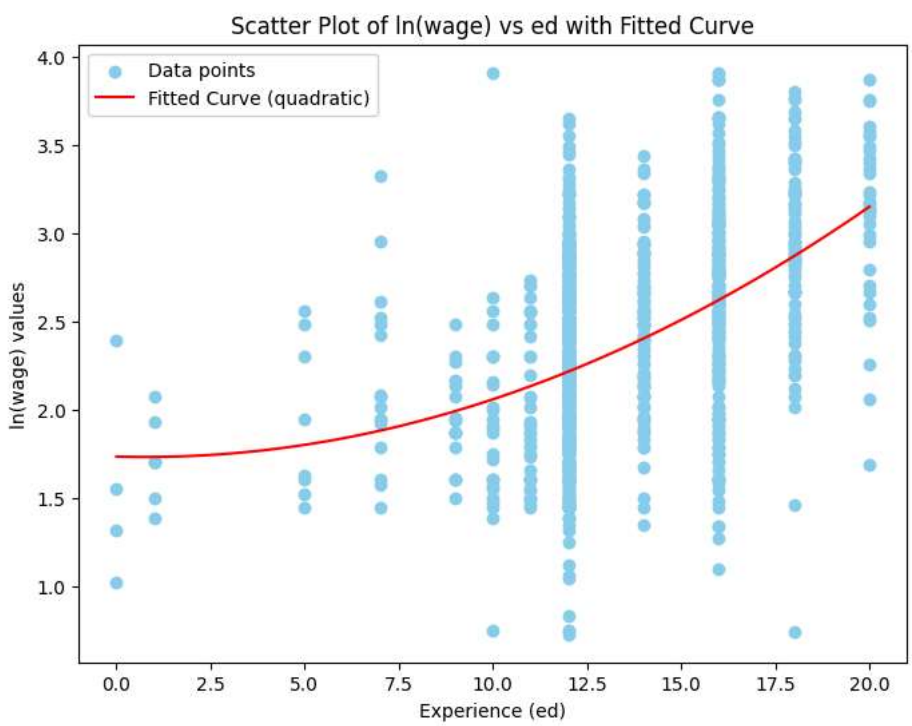
```
reg_result4 = sm.OLS(Y, sm.add_constant(X3)).fit()
print(reg_result4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln(wage)   R-squared:                       0.430
Model:                            OLS   Adj. R-squared:                  0.425
Method:                 Least Squares   F-statistic:                     79.65
Date:                Tue, 24 Sep 2024   Prob (F-statistic):          2.89e-145
Time:                        07:00:19   Log-Likelihood:                -725.89
No. Observations:                1280   AIC:                             1478.
Df Residuals:                    1267   BIC:                             1545.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0023      0.095     10.508      0.000       0.815       1.189
fe            -0.3178      0.137     -2.322      0.020      -0.586      -0.049
nw            -0.1418      0.050     -2.835      0.005      -0.240      -0.044
un             0.1775      0.045      3.986      0.000       0.090       0.265
ed             0.0717      0.007     10.769      0.000       0.059       0.085
ex             0.0360      0.004      9.406      0.000       0.028       0.044
wk             0.2602      0.039      6.626      0.000       0.183       0.337
ex^2          -0.0005   8.27e-05     -6.568      0.000      -0.001      -0.000
fe:nw          0.0460      0.068      0.677      0.498      -0.087       0.179
fe:un          0.0171      0.068      0.251      0.802      -0.117       0.151
fe:ed          0.0116      0.010      1.206      0.228      -0.007       0.030
fe:ex         -0.0035      0.002     -1.609      0.108      -0.008       0.001
fe:wk         -0.0285      0.054     -0.527      0.598      -0.135       0.078
==============================================================================
Omnibus:                       35.547   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.193
Skew:                          -0.159   Prob(JB):                     5.73e-16
```

```
Kurtosis:                       4.102   Cond. No.                    9.24e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Start coding or generate with AI.

```
plt.figure(figsize=(8, 6))
plt.scatter(df1['ed'], df1['ln(wage)'], color='skyblue', label='Data points')

# Fit a second-degree polynomial (quadratic curve)
coefficients = np.polyfit(df1['ed'], df1['ln(wage)'], 2)
polynomial = np.poly1d(coefficients)

# Generate x values (ex range) for plotting the fitted curve
x_vals = np.linspace(df1['ed'].min(), df1['ed'].max(), 100)
y_vals = polynomial(x_vals)

# Plot the fitted curve
plt.plot(x_vals, y_vals, color='red', label='Fitted Curve (quadratic)')

# Add titles and labels
plt.title('Scatter Plot of ln(wage) vs ed with Fitted Curve')
plt.xlabel('Experience (ed)')
plt.ylabel('ln(wage) values')
plt.legend()

# Show the plot
plt.show()
```




```
X4.head()
```

	fe	nw	un	ed	ex	wk	ex^2
0	1.0	0.0	0.0	12.0	20.0	1.0	400.0
1	0.0	0.0	0.0	9.0	9.0	0.0	81.0
2	0.0	0.0	0.0	16.0	15.0	1.0	225.0
3	0.0	1.0	1.0	14.0	38.0	0.0	1444.0
4	1.0	1.0	0.0	16.0	19.0	1.0	361.0

```
X4['ed^2'] = X4['ed']**2
```

```
<ipython-input-149-ee1ac2b9299c>: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-cc
  X4['ed^2'] = X4['ed']**2
```

```
reg_result5 = sm.OLS(Y, sm.add_constant(X3)).fit()
print(reg_result5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                ln(wage)   R-squared:                       0.430
Model:                             OLS   Adj. R-squared:                  0.425
Method:                  Least Squares   F-statistic:                     79.65
Date:                 Tue, 24 Sep 2024   Prob (F-statistic):          2.89e-145
Time:                         07:00:20   Log-Likelihood:                -725.89
No. Observations:                 1280   AIC:                             1478.
Df Residuals:                     1267   BIC:                             1545.
Df Model:                           12
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0023      0.095     10.508      0.000       0.815       1.189
fe            -0.3178      0.137     -2.322      0.020      -0.586      -0.049
nw            -0.1418      0.050     -2.835      0.005      -0.240      -0.044
un             0.1775      0.045      3.986      0.000       0.090       0.265
ed             0.0717      0.007     10.769      0.000       0.059       0.085
ex             0.0360      0.004      9.406      0.000       0.028       0.044
wk             0.2602      0.039      6.626      0.000       0.183       0.337
ex^2          -0.0005   8.27e-05     -6.568      0.000      -0.001      -0.000
fe:nw          0.0460      0.068      0.677      0.498      -0.087       0.179
fe:un          0.0171      0.068      0.251      0.802      -0.117       0.151
fe:ed          0.0116      0.010      1.206      0.228      -0.007       0.030
fe:ex         -0.0035      0.002     -1.609      0.108      -0.008       0.001
fe:wk         -0.0285      0.054     -0.527      0.598      -0.135       0.078
==============================================================================
Omnibus:                       35.547   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.193
Skew:                          -0.159   Prob(JB):                     5.73e-16
Kurtosis:                       4.102   Cond. No.                     9.24e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
from statsmodels.stats.diagnostic import het_breuschpagan

# Predict the values and compute residuals
predicted_vals = reg_result5.fittedvalues
residuals = reg_result5.resid

# Breusch-Pagan Test
lm_test = het_breuschpagan(residuals, sm.add_constant(predicted_vals))
print('Breusch-Pagan p-value:', lm_test[1])
```

```
Breusch-Pagan p-value: 0.0001357136150874963
```

Breusch-Pagan Test:

- How it works: This statistical test assesses **heteroscedasticity** by testing if the residuals' variance is related to the independent variables. If the test is significant, it indicates heteroscedasticity.
- Interpretation: A low p-value (typically < 0.05) indicates the presence of heteroscedasticity.

```
from statsmodels.stats.outliers_influence import reset_ramsey

# Run the OLS regression
model = sm.OLS(df1['ln(wage)'], sm.add_constant(X4[['fe', 'nw', 'un', 'ed', 'ex', 'wk', 'ex^2', 'ed^2']])).fit(

# Perform the Ramsey RESET test
reset_test = reset_ramsey(model)
```

```
print('Ramsey RESET p-value:', reset_test)
```

```
Ramsey RESET p-value: <F test: F=2.2719128983106884, p=0.05955964618806872, df_denom=1.27e+03, df_num=4>
```

OVB If the p-value is small (e.g., less than 0.05), there’s evidence of omitted variables or a misspecified model.

```
model = sm.OLS(df1['ln(wage)'], sm.add_constant(X4[['fe', 'nw', 'un', 'ed', 'ex', 'wk', 'ex^2', 'ed^2']])).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln(wage)   R-squared:                       0.431
Model:                            OLS   Adj. R-squared:                  0.428
Method:                 Least Squares   F-statistic:                     120.6
Date:                Tue, 24 Sep 2024   Prob (F-statistic):         4.88e-150
Time:                        07:45:17   Log-Likelihood:                -724.23
No. Observations:                1280   AIC:                             1466.
Df Residuals:                    1271   BIC:                             1513.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.2848      0.128     10.032      0.000       1.034       1.536
fe            -0.2305      0.024     -9.549      0.000      -0.278      -0.183
nw            -0.1190      0.034     -3.544      0.000      -0.185      -0.053
un             0.1894      0.033      5.664      0.000       0.124       0.255
ed             0.0242      0.018      1.329      0.184      -0.012       0.060
ex             0.0347      0.004      9.658      0.000       0.028       0.042
wk             0.2320      0.027      8.557      0.000       0.179       0.285
ex^2          -0.0006   8.22e-05     -6.826      0.000      -0.001      -0.000
ed^2           0.0021      0.001      3.027      0.003       0.001       0.003
==============================================================================
Omnibus:                       38.729   Durbin-Watson:                   1.977
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               78.616
Skew:                          -0.172   Prob(JB):                     8.49e-18
Kurtosis:                       4.164   Cond. No.                     7.73e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.73e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
# Predict the values and compute residuals
predicted_vals = model.fittedvalues
residuals = model.resid

# Breusch-Pagan Test
lm_test = het_breuschpagan(residuals, sm.add_constant(predicted_vals))
print('Breusch-Pagan p-value:', lm_test[1])
```

```
Breusch-Pagan p-value: 4.1651668328051035e-05
```

```
model2 = sm.OLS(df1['ln(wage)'], sm.add_constant(X4[['fe', 'nw', 'un', 'ex', 'wk', 'ex^2', 'ed^2']])).fit()
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln(wage)   R-squared:                       0.431
Model:                            OLS   Adj. R-squared:                  0.428
Method:                 Least Squares   F-statistic:                     137.5
Date:                Tue, 24 Sep 2024   Prob (F-statistic):         9.64e-151
Time:                        07:45:34   Log-Likelihood:                -725.12
No. Observations:                1280   AIC:                             1466.
Df Residuals:                    1272   BIC:                             1507.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
```