analyze-price-stock

August 12, 2023

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     from google.colab import files
     import io
     import numpy as np
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
       ↪mean_absolute_percentage_error, r2_score
     import warnings

     warnings.filterwarnings('ignore')




     data = files.upload()
```

```
<IPython.core.display.HTML object>

Saving Nat_Gas.csv to Nat_Gas (1).csv
```

```
[ ]: df = pd.read_csv('Nat_Gas.csv')
```

```
[ ]: df
```

```
[ ]:       Dates  Prices
     0   10/31/20   10.10
     1   11/30/20   10.30
     2   12/31/20   11.00
     3    1/31/21   10.90
     4    2/28/21   10.90
     5    3/31/21   10.90
     6    4/30/21   10.40
     7    5/31/21    9.84
     8    6/30/21   10.00
     9    7/31/21   10.10
```

```
10   8/31/21   10.30
11   9/30/21   10.20
12  10/31/21   10.10
13  11/30/21   11.20
14  12/31/21   11.40
15   1/31/22   11.50
16   2/28/22   11.80
17   3/31/22   11.50
18   4/30/22   10.70
19   5/31/22   10.70
20   6/30/22   10.40
21   7/31/22   10.50
22   8/31/22   10.40
23   9/30/22   10.80
24  10/31/22   11.00
25  11/30/22   11.60
26  12/31/22   11.60
27   1/31/23   12.10
28   2/28/23   11.70
29   3/31/23   12.00
30   4/30/23   11.50
31   5/31/23   11.20
32   6/30/23   10.90
33   7/31/23   11.40
34   8/31/23   11.10
35   9/30/23   11.50
36  10/31/23   11.80
37  11/30/23   12.20
38  12/31/23   12.80
39   1/31/24   12.60
40   2/29/24   12.40
41   3/31/24   12.70
42   4/30/24   12.10
43   5/31/24   11.40
44   6/30/24   11.50
45   7/31/24   11.60
46   8/31/24   11.50
47   9/30/24   11.80
```

[ ]:

```
df.head(10)
```

[ ]:

```
      Dates  Prices
0  10/31/20   10.10
1  11/30/20   10.30
2  12/31/20   11.00
3   1/31/21   10.90
4   2/28/21   10.90
```

```
5   3/31/21   10.90
6   4/30/21   10.40
7   5/31/21    9.84
8   6/30/21   10.00
9   7/31/21   10.10
```

[ ]:

```
df.index = pd.to_datetime(df['Dates'])
df
```

[ ]:

```
               Dates  Prices
Dates
2020-10-31  10/31/20   10.10
2020-11-30  11/30/20   10.30
2020-12-31  12/31/20   11.00
2021-01-31   1/31/21   10.90
2021-02-28   2/28/21   10.90
2021-03-31   3/31/21   10.90
2021-04-30   4/30/21   10.40
2021-05-31   5/31/21    9.84
2021-06-30   6/30/21   10.00
2021-07-31   7/31/21   10.10
2021-08-31   8/31/21   10.30
2021-09-30   9/30/21   10.20
2021-10-31  10/31/21   10.10
2021-11-30  11/30/21   11.20
2021-12-31  12/31/21   11.40
2022-01-31   1/31/22   11.50
2022-02-28   2/28/22   11.80
2022-03-31   3/31/22   11.50
2022-04-30   4/30/22   10.70
2022-05-31   5/31/22   10.70
2022-06-30   6/30/22   10.40
2022-07-31   7/31/22   10.50
2022-08-31   8/31/22   10.40
2022-09-30   9/30/22   10.80
2022-10-31  10/31/22   11.00
2022-11-30  11/30/22   11.60
2022-12-31  12/31/22   11.60
2023-01-31   1/31/23   12.10
2023-02-28   2/28/23   11.70
2023-03-31   3/31/23   12.00
2023-04-30   4/30/23   11.50
2023-05-31   5/31/23   11.20
2023-06-30   6/30/23   10.90
2023-07-31   7/31/23   11.40
2023-08-31   8/31/23   11.10
2023-09-30   9/30/23   11.50
```

```
2023-10-31  10/31/23   11.80
2023-11-30  11/30/23   12.20
2023-12-31  12/31/23   12.80
2024-01-31   1/31/24   12.60
2024-02-29   2/29/24   12.40
2024-03-31   3/31/24   12.70
2024-04-30   4/30/24   12.10
2024-05-31   5/31/24   11.40
2024-06-30   6/30/24   11.50
2024-07-31   7/31/24   11.60
2024-08-31   8/31/24   11.50
2024-09-30   9/30/24   11.80
```

[ ]:

```
splitted = df['Dates'].str.split('/' , expand=True)
df['days'] = splitted[1].astype('int')
df['month'] = splitted[0].astype('int')
df['year'] = splitted[2].astype('int')
df.head()
```

[ ]:

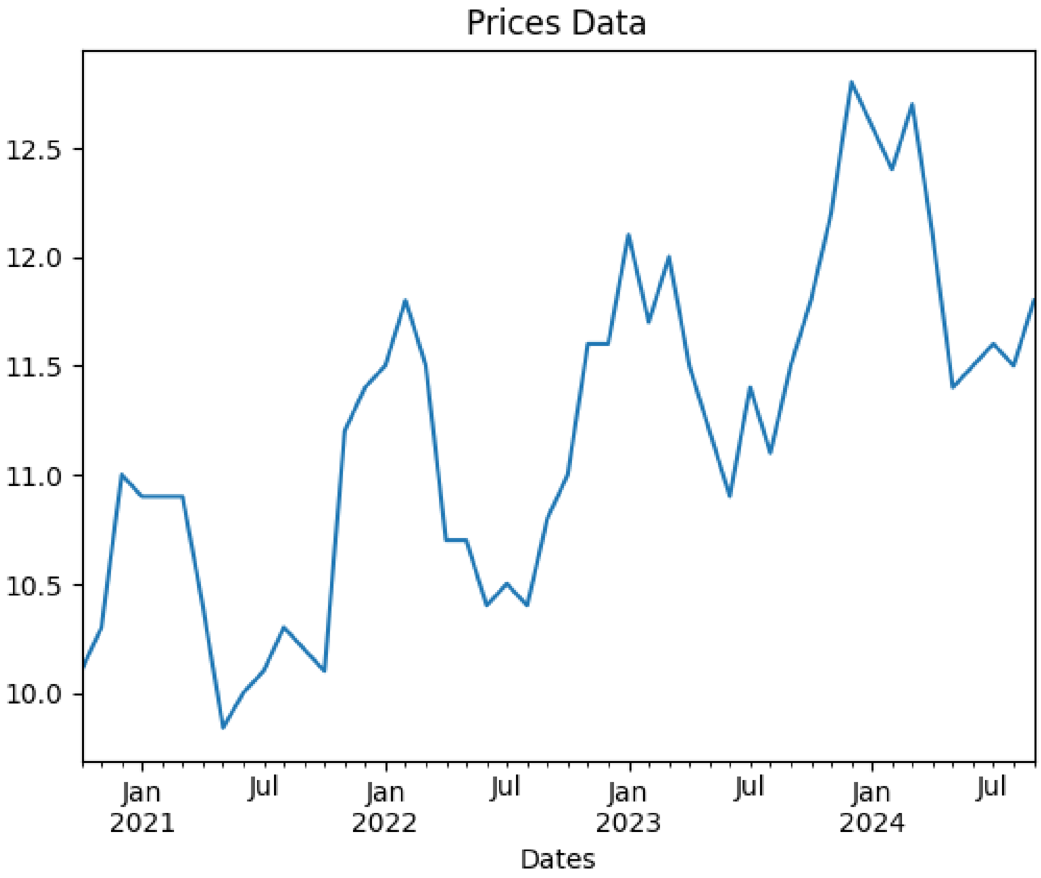
```
               Dates  Prices  days  month  year
Dates
2020-10-31  10/31/20    10.1    31     10    20
2020-11-30  11/30/20    10.3    30     11    20
2020-12-31  12/31/20    11.0    31     12    20
2021-01-31   1/31/21    10.9    31      1    21
2021-02-28   2/28/21    10.9    28      2    21
```

[ ]:

```
plt.title('Prices Data')
df['Prices'].plot()
```

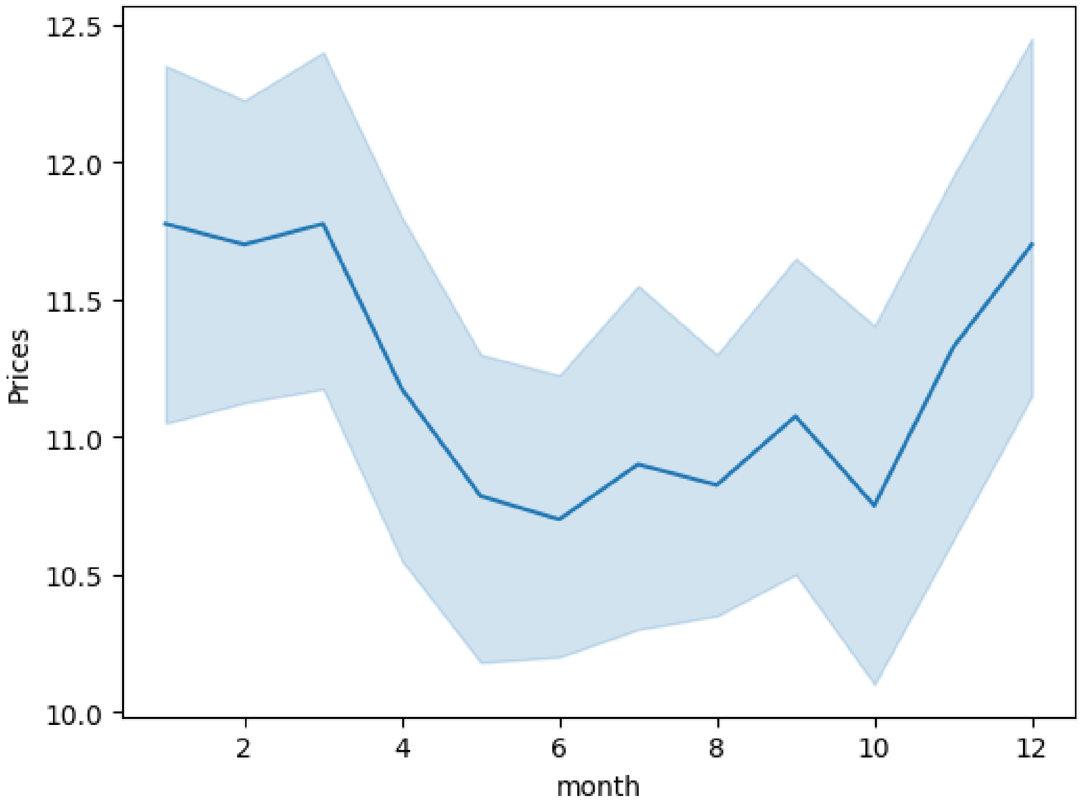
[ ]:

```
<Axes: title={'center': 'Prices Data'}, xlabel='Dates'>
```



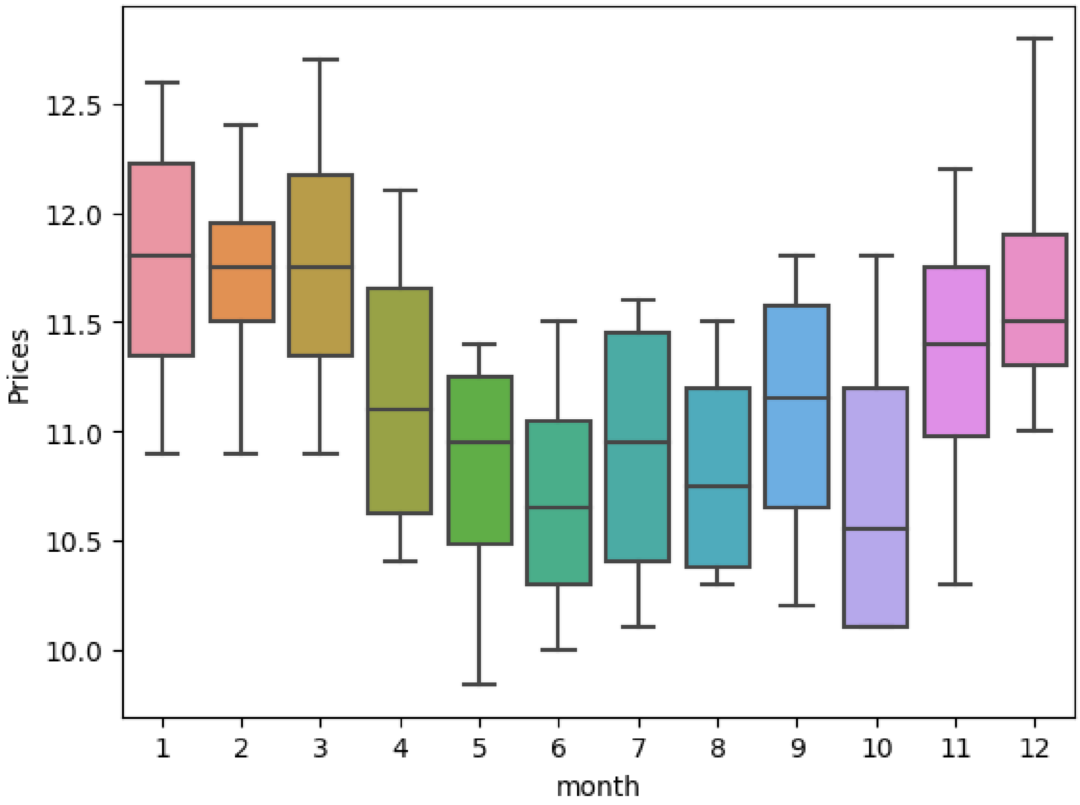
```
[ ]: sns.lineplot(x='month' , y= 'Prices' , data=df)
```

```
[ ]: <Axes: xlabel='month', ylabel='Prices'>
```



```
[ ]: sns.boxplot(x='month', y='Prices' ,data=df)
```

```
[ ]: <Axes: xlabel='month', ylabel='Prices'>
```

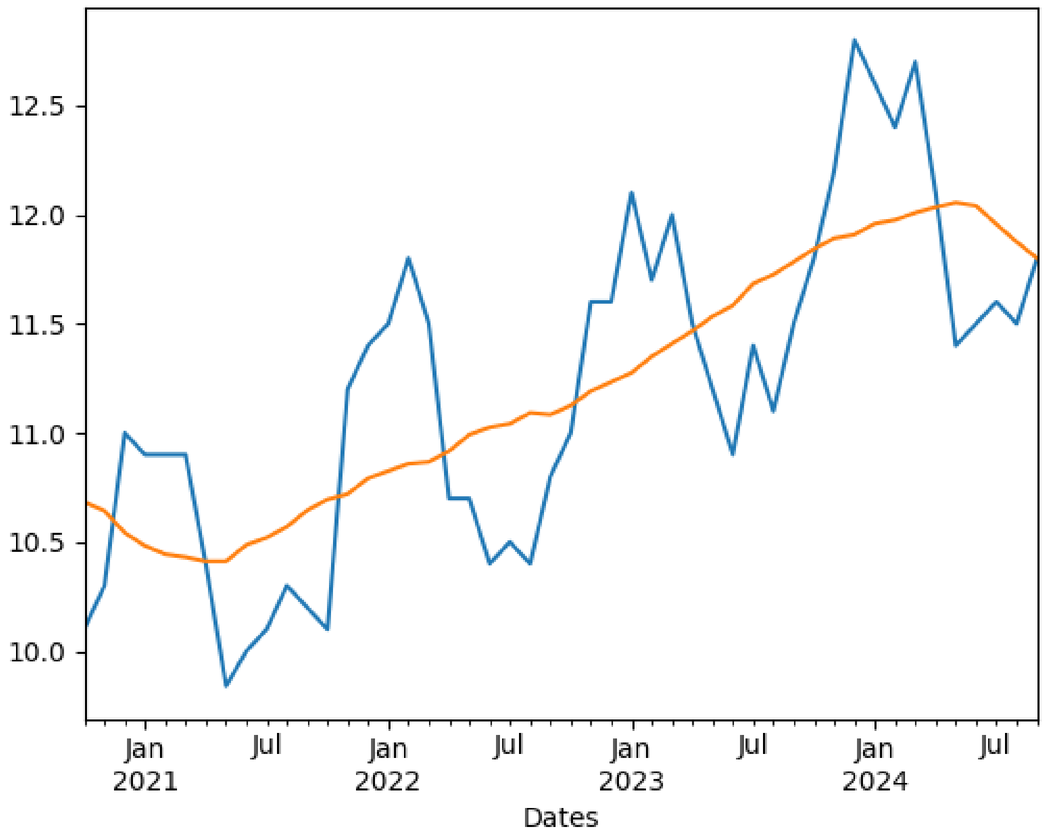


```
[ ]: from statsmodels.tsa.stattools import adfuller
     result = adfuller(df['Prices'],autolag='AIC')

     print(f'ADF statistics: {result[0]}')
     print(f'p_value: {result[1]}')

     for key, value in result[4].items():
       print(f'Critical value {key}: {value}')
```

```
ADF statistics: 0.218076861699999427
p_value: 0.973257438844869
Critical value 1%: -3.6209175221605827
Critical value 5%: -2.9435394610388332
Critical value 10%: -2.6104002410518627
```

```
[ ]: ma = df['Prices'].rolling(window=12, center=True,min_periods=6).mean()
     ax = df['Prices'].plot()
     ma.plot(ax=ax)
```

```
[ ]: <Axes: xlabel='Dates'>
```



```
[ ]: X = df['year']
     y = df['Prices']
```

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7,␣
       ↪test_size= 0.3, random_state= 100 )
```

```
[ ]: X_train.head()
```

```
[ ]: Dates
     2023-09-30    23
     2023-07-31    23
     2021-05-31    21
     2024-07-31    24
     2022-06-30    22
     Name: year, dtype: int64
```

```
[ ]: y_train.head()
```

```
[ ]: Dates
     2023-09-30    11.50
```

```
2023-07-31    11.40
2021-05-31     9.84
2024-07-31    11.60
2022-06-30    10.40
Name: Prices, dtype: float64
```

```
[ ]: import statsmodels.api as sm
```

```
[ ]: X_train_sm = sm.add_constant(X_train)
     lr = sm.OLS(y_train, X_train_sm).fit()
```

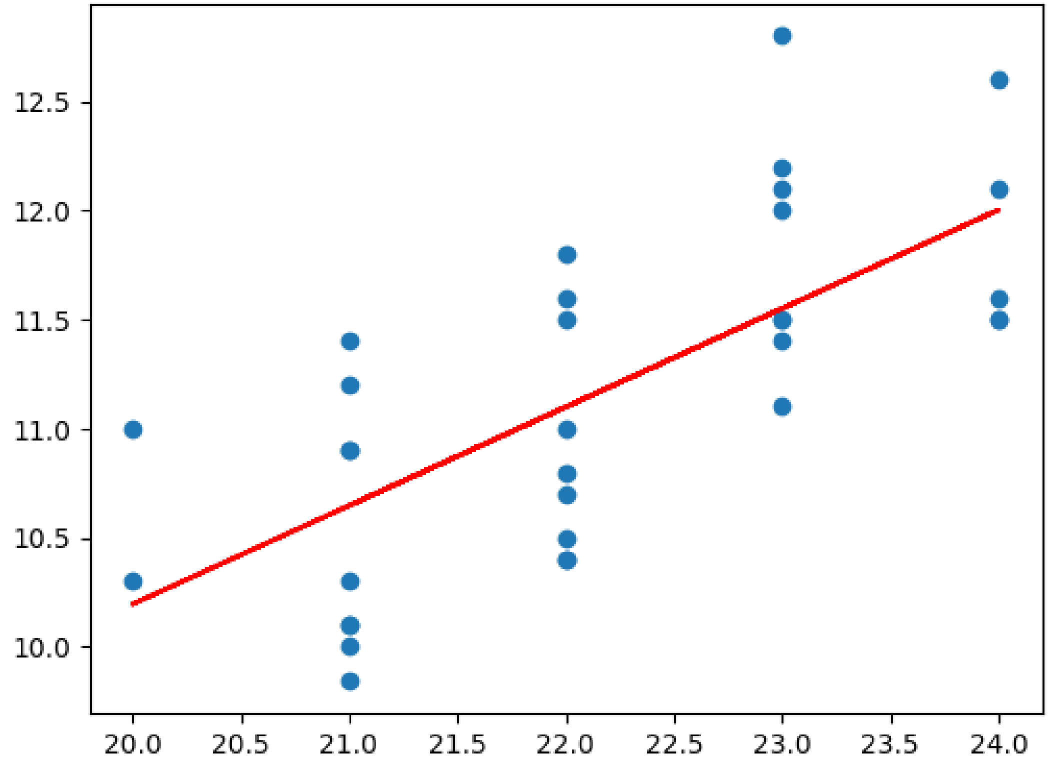
```
[ ]: lr.params
```

```
[ ]: const    1.154479
     year     0.452178
     dtype: float64
```

```
[ ]: print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Prices   R-squared:                       0.483
Model:                            OLS   Adj. R-squared:                  0.466
Method:                 Least Squares   F-statistic:                     28.92
Date:                Mon, 26 Jun 2023   Prob (F-statistic):           7.26e-06
Time:                        14:55:00   Log-Likelihood:                -26.615
No. Observations:                  33   AIC:                             57.23
Df Residuals:                      31   BIC:                             60.22
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.1545      1.865      0.619      0.540      -2.649       4.958
year           0.4522      0.084      5.378      0.000       0.281       0.624
==============================================================================
Omnibus:                        3.819   Durbin-Watson:                   2.609
Prob(Omnibus):                  0.148   Jarque-Bera (JB):                2.005
Skew:                           0.325   Prob(JB):                        0.367
Kurtosis:                       1.982   Cond. No.                         426.
==============================================================================

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

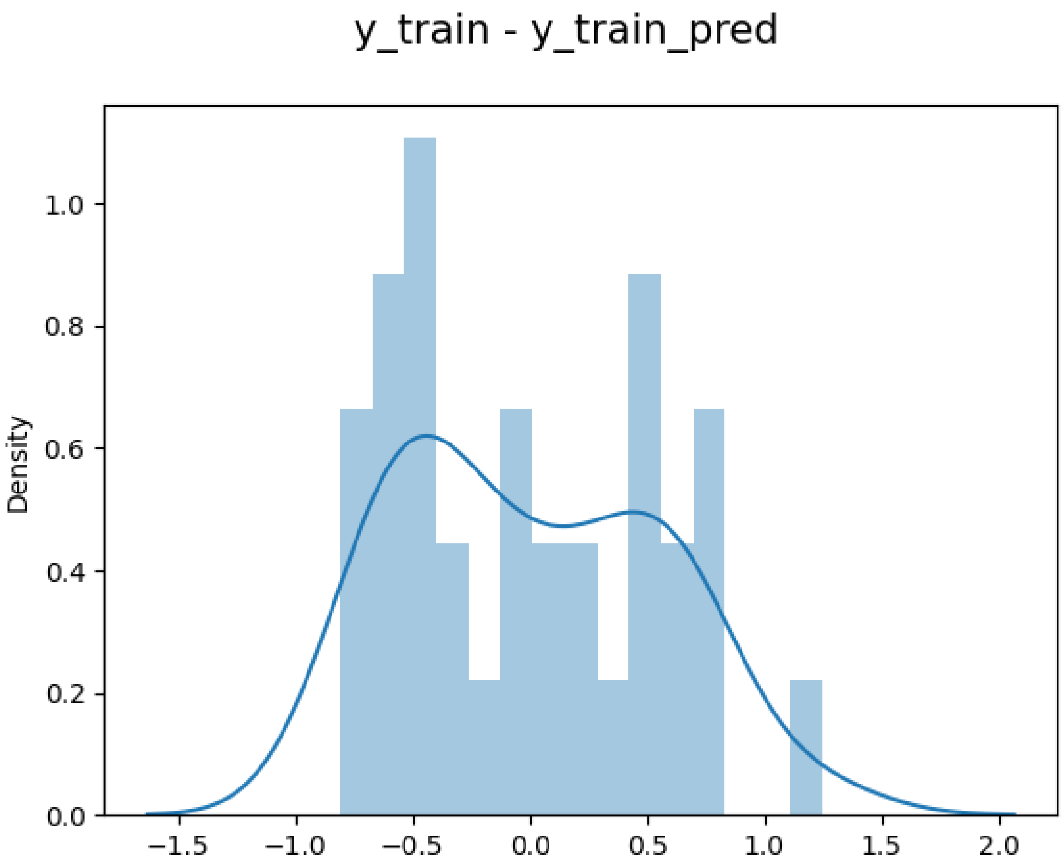
```
[ ]: plt.scatter(X_train, y_train)
     plt.plot(X_train, 1.154 + 0.452*X_train, 'r')
     plt.show()
```



```
[ ]: y_train_pred = lr.predict(X_train_sm)
     res = (y_train - y_train_pred)
```

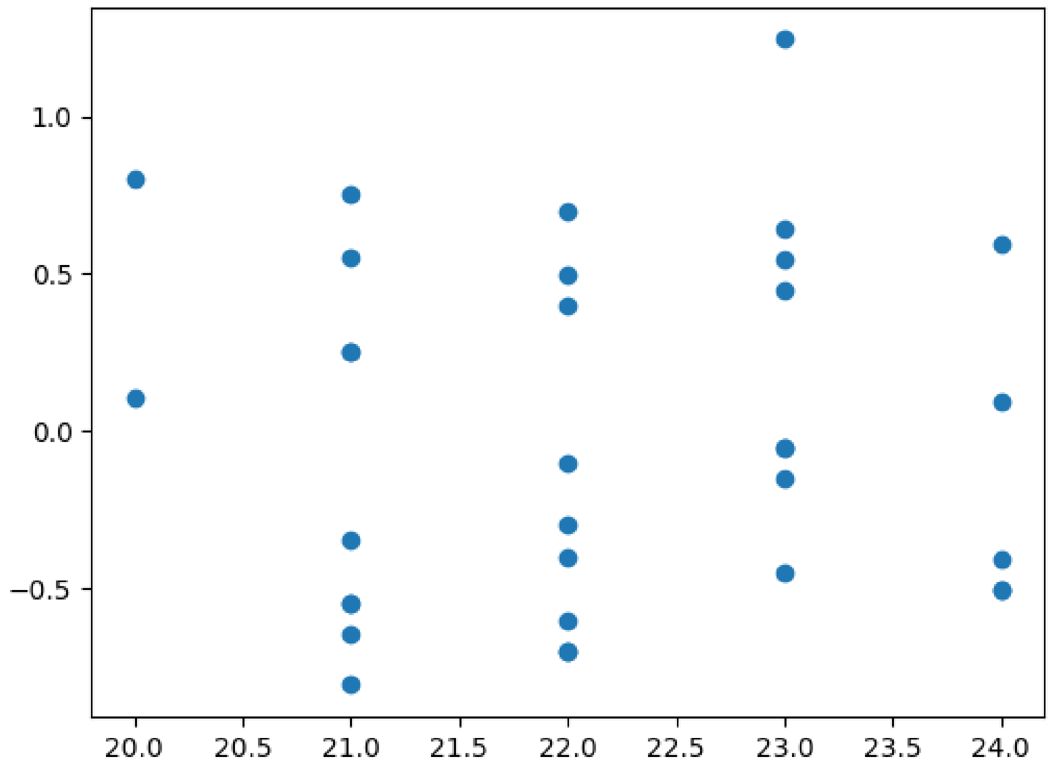
```
[ ]: fig = plt.figure ()
     sns.distplot(res, bins = 15)
     fig.suptitle('y_train - y_train_pred', fontsize = 15)
     plt.show
```

```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```



[ ]:

```
plt.scatter(X_train,res)
plt.show()
```



[ ]:
```
X_test_sm = sm.add_constant(X_test)
y_pred = lr.predict(X_test_sm)
```

[ ]:
```
y_pred.head()
```

[ ]:
```
Dates
2021-04-30    10.650219
2023-02-28    11.554575
2023-06-30    11.554575
2022-12-31    11.102397
2024-03-31    12.006753
dtype: float64
```

[ ]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

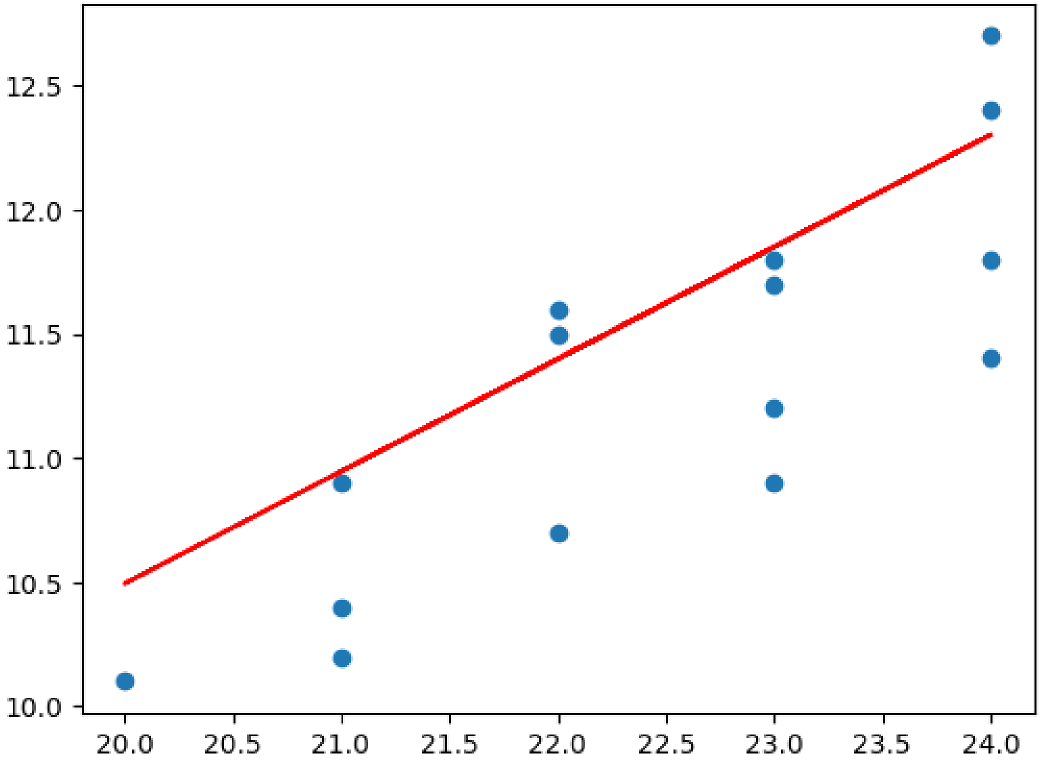
[ ]:
```
np.sqrt(mean_squared_error(y_test, y_pred))
```

[ ]:
```
0.4152616819427225
```

[ ]:
```
r_squared = r2_score(y_test, y_pred)
r_squared
```

```
[ ]: 0.6813443699882783
```

```
[ ]: plt.scatter(X_test, y_test)
     plt.plot(X_test, 1.454 +0.452*X_test , 'r')
     plt.show()
```


12.5
12.0
11.5
11.0
10.5
10.0
20.0
20.5
21.0
21.5
22.0
22.5
23.0
23.5
24.0