In [134…
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
```

0: Data preparation

Note: the first row, empty columns and text boxes were **deleted before proccessing in python**

In [135…
```python
data = pd.read_excel("PG DS - Middle Econometrist Test task.xlsx", sheet_name="data")
print(data.columns)
data.describe()
```

```
Index(['Date', 'Year', 'Week', 'MS_value_%', 'MS_volume_%',
       'Weighted_distribution', 'Volume sales, pcs', 'Value sales, UAH',
       'Video, TRP', 'SMM, TRP', 'TV, TRP'],
      dtype='object')
```

Out[135…

	Year	Week	MS_value_%	MS_volume_%	Weighted_distribution	Volume sales, pcs	Value sales, UAH	Video, TRP	SMM, TRP	TV, TRP
count	167.000000	167.000000	167.000000	167.000000	167.000000	1.670000e+02	1.670000e+02	167.000000	167.000000	167.000000
mean	2021.131737	25.149701	0.177262	0.075579	89.461317	4.150532e+05	3.562811e+07	5.236311	13.134937	102.835310
std	0.934937	15.438814	0.019472	0.011475	6.208531	2.519513e+05	2.164018e+07	6.947833	11.686159	108.845697
min	2020.000000	1.000000	0.135683	0.051484	69.410000	6.695836e+04	4.749757e+06	0.000000	0.000000	0.000000
25%	2020.000000	11.000000	0.166401	0.067730	85.750000	2.079299e+05	1.652133e+07	0.000000	0.000000	0.000000
50%	2021.000000	25.000000	0.175959	0.074738	92.270000	3.787262e+05	3.659004e+07	0.000000	14.677845	0.000000
75%	2022.000000	38.500000	0.185486	0.081772	94.020000	5.814479e+05	4.833492e+07	9.736477	20.624273	209.840000
max	2023.000000	52.000000	0.242038	0.107712	95.260000	1.118501e+06	1.047959e+08	32.284152	47.860061	311.550000

In [136…
```python
try:
    data["Date"] = data["Date"].str.replace(" ", "-")
    data["Date"] = pd.to_datetime(data["Date"] + "-1", format="%Y-W%W-%w")
except Exception as e:
    pass
print(data.info())
data.head(3)
```

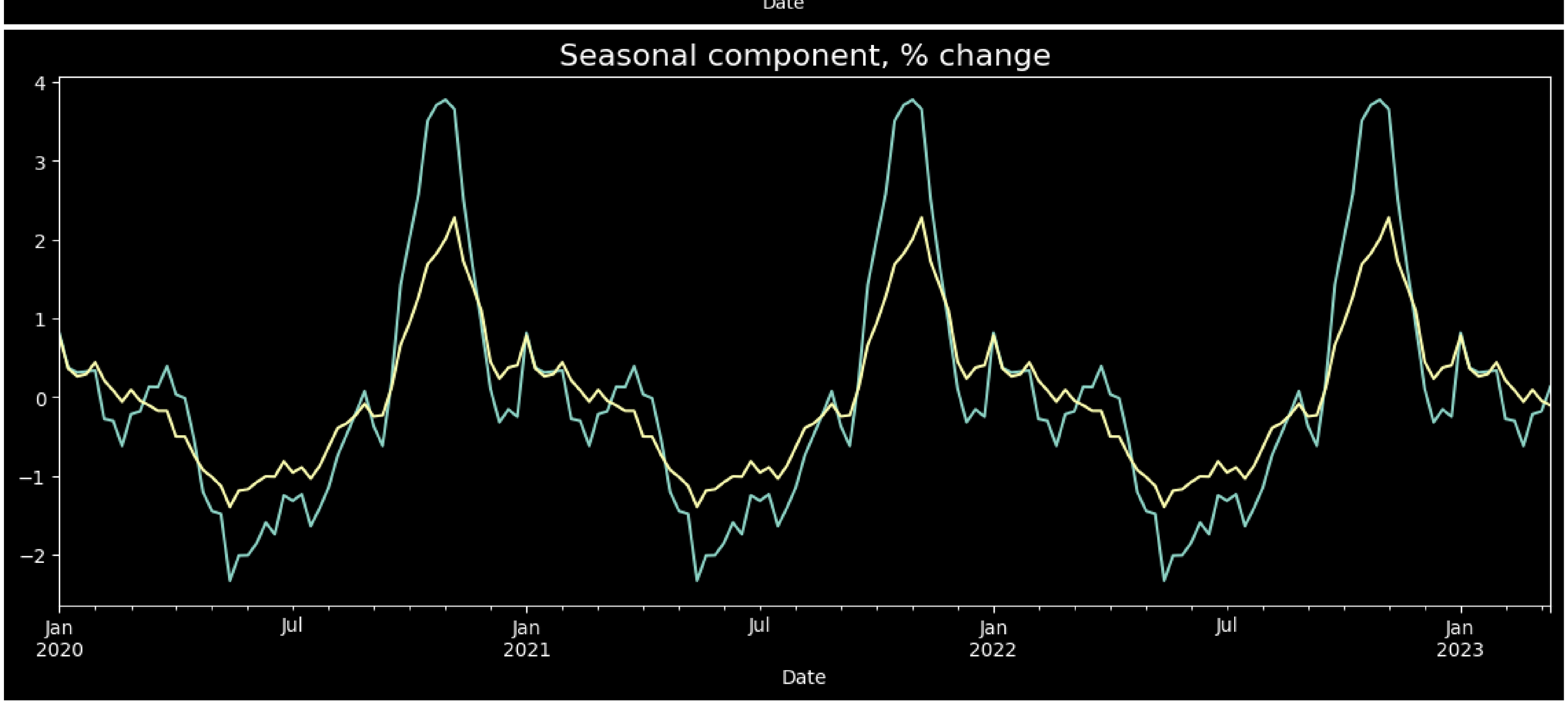
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Date                   167 non-null    datetime64[ns]
 1   Year                   167 non-null    int64
 2   Week                   167 non-null    int64
 3   MS_value_%             167 non-null    float64
 4   MS_volume_%            167 non-null    float64
 5   Weighted_distribution  167 non-null    float64
 6   Volume sales, pcs      167 non-null    float64
 7   Value sales, UAH       167 non-null    float64
 8   Video, TRP             167 non-null    float64
 9   SMM, TRP               167 non-null    float64
 10  TV, TRP                167 non-null    float64
dtypes: datetime64[ns](1), float64(8), int64(2)
memory usage: 14.5 KB
None
```

Out[136…

	Date	Year	Week	MS_value_%	MS_volume_%	Weighted_distribution	Volume sales, pcs	Value sales, UAH	Video, TRP	SMM, TRP	TV, TRP
0	2020-01-06	2020	1	0.194662	0.073238	92.99	684745.27	46790938.18	0.000000	0.0	206.18
1	2020-01-13	2020	2	0.194925	0.071813	92.99	621840.71	41667580.50	0.000000	0.0	222.20
2	2020-01-20	2020	3	0.186264	0.067827	92.99	671657.94	44403164.56	9.548402	0.0	201.56

1. Analyze brand's volume & value MS% dynamics, estimate brand's seasonality

In [137…
```python
temp = data.copy()
temp["MS_value_%"], temp["MS_volume_%"] = temp["MS_value_%"] * 100, temp["MS_volume_%"] * 100

plt.style.use('dark_background')

fig, ax = plt.subplots(figsize=(14, 7))
ax2 = ax.twinx()
sns.lineplot(data=temp, x="Date", y="MS_value_%", marker="o", ax=ax, color="cyan", markersize=2.5, linewidth=2)
sns.lineplot(data=temp, x="Date", y="MS_volume_%", marker="o", ax=ax2, color="white", markersize=2.5, linewidth=2)
ax.set_ylabel("MS value, %", color="cyan", fontsize=14)
ax.tick_params(axis='y', colors='cyan')
ax2.set_ylabel("MS volume, %", color="white", fontsize=14)
plt.title("Market Share Value and Volume Over Time", fontsize=16)
# sns.despine(left=True, right=True)
plt.show()

decomposition_value = seasonal_decompose(temp.set_index("Date")["MS_value_%"], model='additive', period=52)
decomposition_volume = seasonal_decompose(temp.set_index("Date")["MS_volume_%"], model='additive', period=52)
decomposition_value.seasonal.plot(figsize=(14, 5), title="Seasonal Component of MS_value_%")
decomposition_volume.seasonal.plot(figsize=(14, 5), title="Seasonal Component of MS_volume_%")
plt.title("Seasonal component, % change", fontsize=16)
plt.show()
```



2. Calculate average brand price, price index (brand's price relatively to market price).

In [138…
```python
data["brand_volume"] = data["Volume sales, pcs"] * data["MS_volume_%"]
data["brand_value"] = data["Value sales, UAH"] * data["MS_value_%"]

data["avg_brand_price"] = data["brand_value"] / data["brand_volume"]
data["market_avg_price"] = data["Value sales, UAH"] / data["Volume sales, pcs"]

data["price_index"] = data["avg_brand_price"] / data["market_avg_price"]

pd.DataFrame(data.agg({"avg_brand_price": "mean",
                       "market_avg_price": "mean",
                       "price_index": "mean"}), columns=["Value"]).T.rename(columns={"avg_brand_price": "Average Brand Price",
                                                                                      "market_avg_price": "Average Market Price",
                                                                                      "price_index": "Average Price Index"})
```

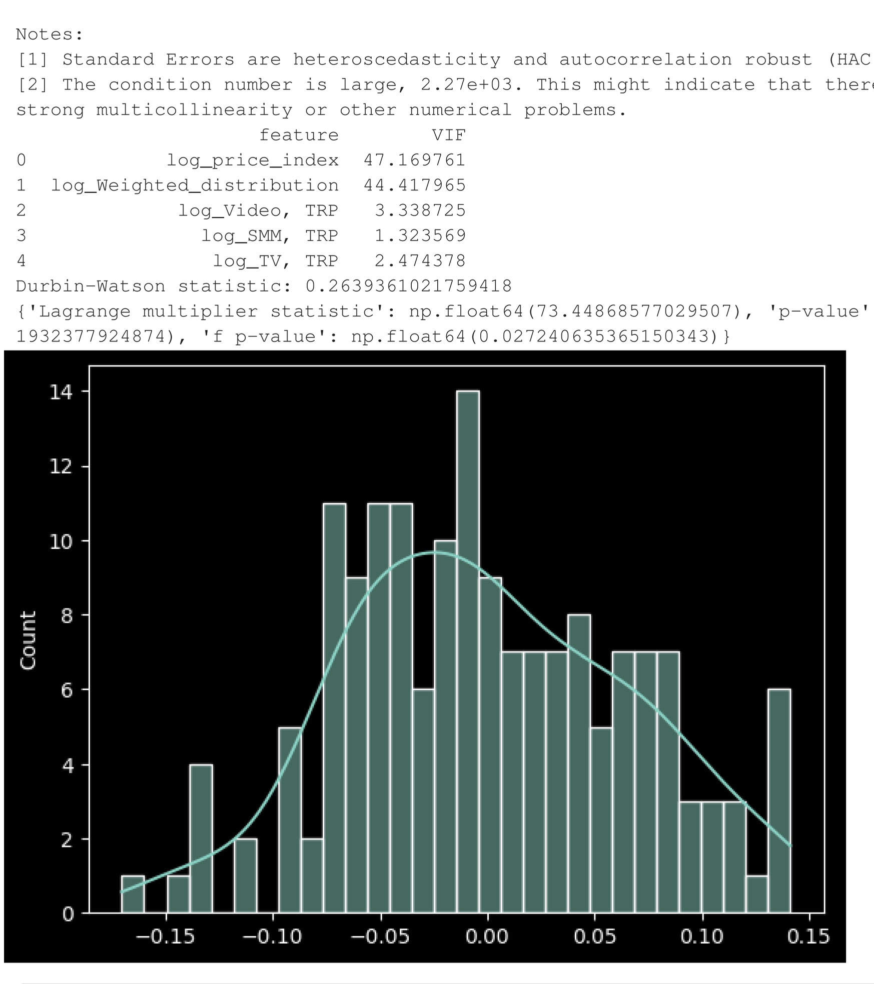
Out[138…

	Average Brand Price	Average Market Price	Average Price Index
Value	201.556362	86.496892	2.378572

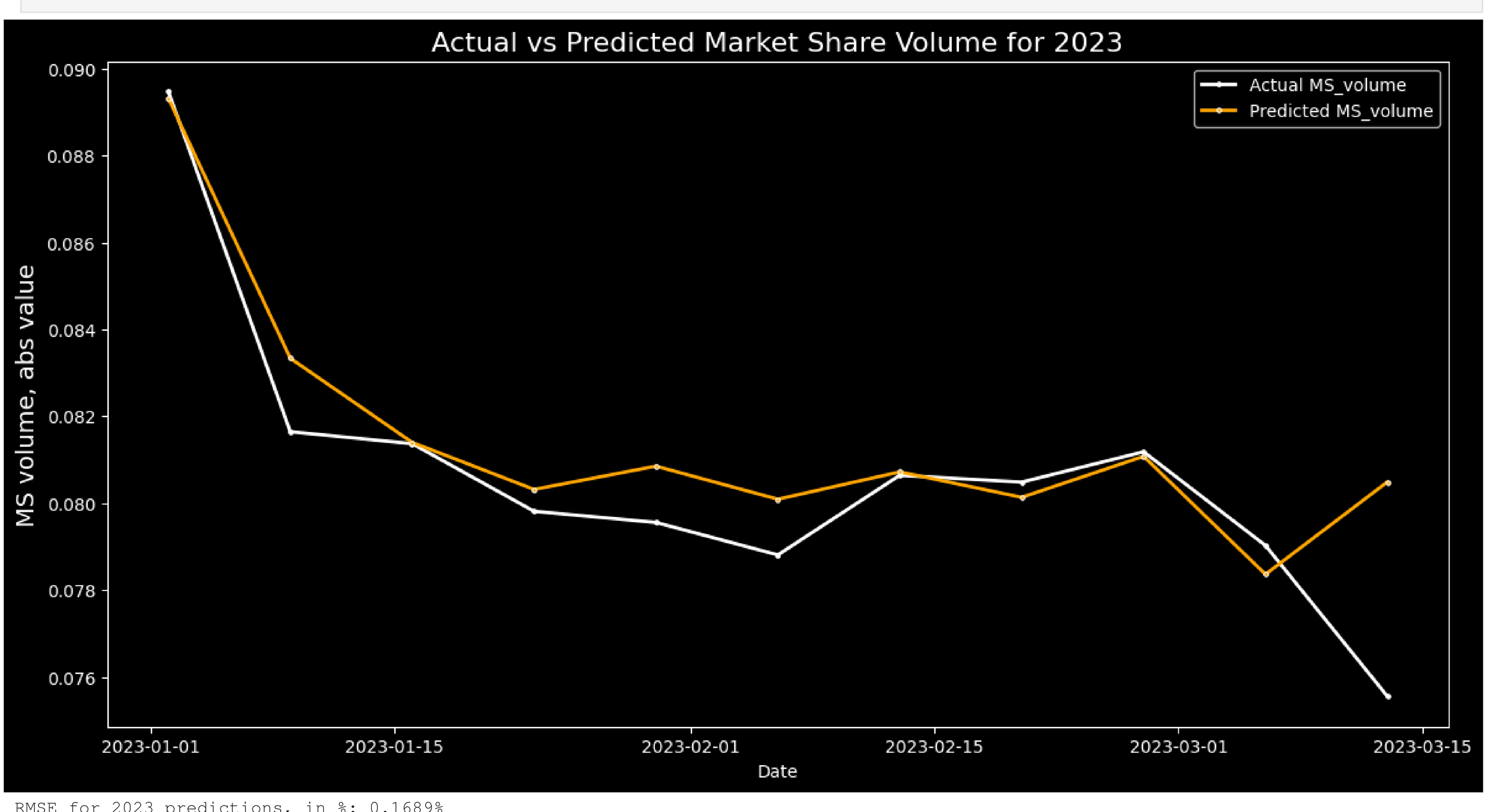
3. Assuming all available variables are exogenous, build a regression model to estimate influence of price, weighted distribution & media support on volume MS%. Include seasonality, if relevant. Please justify the chosen regression method.

In [139…
```python
# data["media_support, TRP"] = data[["Video, TRP", "SMM, TRP", "TV, TRP"]].sum(axis=1)
week_dummies = pd.get_dummies(data["Week"], prefix="week", drop_first=True).astype(int)
for col in ["price_index", "Weighted_distribution", "Video, TRP", "SMM, TRP", "TV, TRP"]:
    log_name = "log_" + col
    data[log_name] = np.log(data[col] + 1e-6)
y = np.log(data["MS_volume_%"])
X = data[["log_price_index", "log_Weighted_distribution", "log_Video, TRP", "log_SMM, TRP", "log_TV, TRP"]].copy()

X1 = sm.add_constant(X)
model1 = sm.OLS(y, X1).fit(cov_type="HAC", cov_kwds={"maxlags": 8})

X2 = sm.add_constant(pd.concat([X, week_dummies], axis=1))
model2 = sm.OLS(y, X2).fit(cov_type='HAC', cov_kwds={"maxlags": 8})

best = model1 if model1.rsquared_adj > model2.rsquared_adj else model2
print(best.summary())

from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)

from statsmodels.stats.stattools import durbin_watson
dw_stat = durbin_watson(best.resid)
print(f'Durbin-Watson statistic: {dw_stat}')

import statsmodels.stats.api as sms
name = ['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value']
test = sms.het_breuschpagan(best.resid, best.model.exog)
print(dict(zip(name, test)))

sns.histplot(best.resid, bins=30, kde=True)
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            MS_volume_%   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.707
Method:                 Least Squares   F-statistic:                     15.75
Date:                Fri, 21 Nov 2025   Prob (F-statistic):           4.55e-33
Time:                        22:07:07   Log-Likelihood:                 215.59
No. Observations:                 167   AIC:                            -317.2
Df Residuals:                     110   BIC:                            -139.4
Df Model:                          56
Covariance Type:                  HAC
=============================================================================================
                                coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------------
const                        -2.4454      1.463     -1.671      0.095      -5.313       0.423
log_price_index              -0.4522      0.153     -2.947      0.003      -0.753      -0.151
log_Weighted_distribution     0.0732      0.302      0.242      0.809      -0.519       0.666
log_Video, TRP                0.0010      0.002      0.620      0.535      -0.002       0.004
log_SMM, TRP                 -0.0003      0.002     -0.191      0.848      -0.004       0.003
log_TV, TRP                  -0.0011      0.001     -0.737      0.461      -0.004       0.002
week_2                       -0.0411      0.015     -2.722      0.006      -0.071      -0.012
week_3                       -0.0627      0.022     -2.786      0.005      -0.107      -0.019
week_4                       -0.0708      0.028     -2.517      0.012      -0.126      -0.016
week_5                       -0.0732      0.039     -1.883      0.060      -0.149       0.003
week_6                       -0.0885      0.042     -2.097      0.036      -0.171      -0.006
week_7                       -0.0824      0.036     -2.315      0.021      -0.152      -0.013
week_8                       -0.0913      0.025     -3.608      0.000      -0.141      -0.042
week_9                       -0.0796      0.021     -3.791      0.000      -0.121      -0.038
week_10                      -0.0935      0.025     -3.704      0.000      -0.143      -0.044
week_11                      -0.0893      0.033     -2.736      0.006      -0.153      -0.025
week_12                      -0.0526      0.052     -1.003      0.316      -0.155       0.050
week_13                      -0.0803      0.041     -1.978      0.048      -0.160      -0.001
week_14                      -0.1312      0.038     -3.451      0.001      -0.206      -0.057
week_15                      -0.1728      0.055     -3.119      0.002      -0.281      -0.064
week_16                      -0.1737      0.042     -4.112      0.000      -0.256      -0.091
week_17                      -0.2114      0.049     -4.321      0.000      -0.307      -0.115
week_18                      -0.2335      0.042     -5.514      0.000      -0.316      -0.150
week_19                      -0.2456      0.049     -4.989      0.000      -0.342      -0.149
week_20                      -0.2722      0.051     -5.307      0.000      -0.373      -0.172
week_21                      -0.2524      0.054     -4.638      0.000      -0.359      -0.146
week_22                      -0.2487      0.054     -4.601      0.000      -0.355      -0.143
week_23                      -0.2401      0.064     -3.777      0.000      -0.365      -0.115
week_24                      -0.2107      0.065     -3.227      0.001      -0.339      -0.083
week_25                      -0.1879      0.061     -3.076      0.002      -0.308      -0.068
week_26                      -0.1759      0.063     -2.811      0.005      -0.298      -0.053
week_27                      -0.1777      0.054     -3.296      0.001      -0.283      -0.072
week_28                      -0.1687      0.046     -3.654      0.000      -0.259      -0.078
week_29                      -0.1931      0.046     -4.222      0.000      -0.283      -0.103
week_30                      -0.1726      0.044     -3.965      0.000      -0.258      -0.087
week_31                      -0.1513      0.046     -3.311      0.001      -0.241      -0.062
week_32                      -0.1240      0.037     -3.334      0.001      -0.197      -0.051
week_33                      -0.1130      0.042     -2.672      0.008      -0.196      -0.030
week_34                      -0.0977      0.037     -2.629      0.009      -0.171      -0.025
week_35                      -0.0789      0.034     -2.346      0.019      -0.145      -0.013
week_36                      -0.0956      0.035     -2.699      0.007      -0.165      -0.026
week_37                      -0.1000      0.024     -4.230      0.000      -0.146      -0.054
week_38                      -0.0525      0.029     -1.779      0.075      -0.110       0.005
week_39                      -0.0205      0.042     -0.485      0.627      -0.103       0.062
week_40                       0.0021      0.044      0.048      0.961      -0.084       0.088
week_41                       0.0170      0.056      0.306      0.759      -0.092       0.126
week_42                       0.0491      0.064      0.770      0.441      -0.076       0.174
week_43                       0.0463      0.073      0.635      0.525      -0.097       0.189
week_44                       0.0765      0.066      1.168      0.243      -0.052       0.205
week_45                       0.1001      0.069      1.452      0.146      -0.035       0.235
week_46                       0.0658      0.051      1.279      0.201      -0.035       0.166
week_47                       0.0463      0.044      1.047      0.295      -0.040       0.133
week_48                       0.0416      0.053      0.791      0.429      -0.061       0.145
week_49                       0.0295      0.073      0.406      0.685      -0.113       0.172
week_50                       0.0170      0.073      0.231      0.817      -0.127       0.161
week_51                       0.0259      0.070      0.370      0.711      -0.111       0.163
week_52                       0.0058      0.056      0.104      0.917      -0.103       0.115
==============================================================================
Omnibus:                        2.052   Durbin-Watson:                   0.264
Prob(Omnibus):                  0.358   Jarque-Bera (JB):                1.686
Skew:                           0.094   Prob(JB):                        0.430
Kurtosis:                       2.545   Cond. No.                     2.27e+03
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 8 lags and without small sample correction
[2] The condition number is large, 2.27e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
                     feature        VIF
0            log_price_index  47.169761
1  log_Weighted_distribution  44.417965
2             log_Video, TRP   3.338725
3               log_SMM, TRP   1.323569
4                log_TV, TRP   2.474378
Durbin-Watson statistic: 0.2639361021759418
{'Lagrange multiplier statistic': np.float64(73.44868577029507), 'p-value': np.float64(0.05882403204064099), 'f-value': np.float64(1.542
1932377924874), 'f p-value': np.float64(0.027240635365150343)}
```



In [141…
```python
df_2023 = data[data["Year"] == 2023].copy()
# df_2023["media_support, TRP"] = df_2023[["Video, TRP", "SMM, TRP", "TV, TRP"]].sum(axis=1)
for col in ["price_index", "Weighted_distribution", "Video, TRP", "SMM, TRP", "TV, TRP"]:
    log_name = "log_" + col
    df_2023[log_name] = np.log(df_2023[col] + 1e-6)
week_dummies_2023 = pd.get_dummies(df_2023["Week"], prefix="week", drop_first=True).astype(int)
for col in week_dummies.columns:
    if col not in week_dummies_2023:
        week_dummies_2023[col] = 0
week_dummies_2023 = week_dummies_2023[week_dummies.columns]  # same order
X_2023 = df_2023[["log_price_index", "log_Weighted_distribution", "log_Video, TRP", "log_SMM, TRP", "log_TV, TRP"]].copy()
X_2023 = sm.add_constant(pd.concat([X_2023, week_dummies_2023], axis=1))

df_2023["predicted_log_MS_volume_%"] = best.predict(X_2023)
df_2023["predicted_MS_volume_%"] = np.exp(df_2023["predicted_log_MS_volume_%"])
df_2023[["Date", "MS_volume_%", "predicted_MS_volume_%"]]

# plot actual vs predicted
fig, ax = plt.subplots(figsize=(14, 7))
sns.lineplot(data=df_2023, x="Date", y="MS_volume_%", marker="o", ax=ax, color="white", markersize=2.5, linewidth=2, label="Actual MS_v
sns.lineplot(data=df_2023, x="Date", y="predicted_MS_volume_%", marker="o", ax=ax, color="orange", markersize=2.5, linewidth=2, label="
ax.set_ylabel("MS volume, abs value", color="white", fontsize=14)
ax.tick_params(axis='y', colors='white')
plt.title("Actual vs Predicted Market Share Volume for 2023", fontsize=16)
plt.legend()
plt.show()

# rmse
rmse = np.sqrt(np.mean((df_2023["MS_volume_%"] - df_2023["predicted_MS_volume_%"]) ** 2))
print(f'RMSE for 2023 predictions, in %: {rmse*100:.4f}%')
```




```
RMSE for 2023 predictions, in %: 0.1689%
```