exercise-module-53

October 30, 2024

1 Module 53: Regression models

Author: Juliho Castillo Colmenares Ph.D.

[75]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.stats.diagnostic import het_white
from statsmodels.stats.stattools import jarque_bera, durbin_watson
from scipy import stats
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

[76]:
```
# Load the data
data = pd.read_csv("advertising.csv")
data.head()
```

[76]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

[77]:
```
# Define features and target variable
X = data[["TV", "Radio", "Newspaper"]]
y = data["Sales"]
```

[78]:
```
# Split data into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=1)
```

[79]:
```
# Add a constant to the model (intercept)
X_train_const = sm.add_constant(X_train)
```

[80]:
```
# Fit the initial model
model = sm.OLS(y_train, X_train_const).fit()
```

[81]:
```python
# Stepwise selection process
significant_features = list(X.columns)
significant_features
```

[81]: ['TV', 'Radio', 'Newspaper']

[82]:
```python
while True:
    model = sm.OLS(y_train, sm.add_constant(X_train[significant_features])).
↪fit()
    p_values = model.pvalues.iloc[1:]  # Exclude intercept
    max_p_value = p_values.max()
    if max_p_value > 0.05:
        excluded_feature = p_values.idxmax()
        significant_features.remove(excluded_feature)
    else:
        break
```

[83]:
```python
# Final model with significant features only
X_train_final = X_train[significant_features]
X_train_final_const = sm.add_constant(X_train_final)
final_model = sm.OLS(y_train, X_train_final_const).fit()
```

[84]:
```python
# Evaluate model on test data
X_test_final = X_test[significant_features]
X_test_final_const = sm.add_constant(X_test_final)
y_pred = final_model.predict(X_test_final_const)
```

[85]:
```python
# Calculate R-squared, Adjusted R-squared, MSE, and MAE
r_squared = final_model.rsquared
adjusted_r_squared = final_model.rsquared_adj
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"R-squared: {r_squared}")
print(f"Adjusted R-squared: {adjusted_r_squared}")
print(f"MSE: {mse}")
print(f"MAE: {mae}")
```

```
R-squared: 0.8993439479032659
Adjusted R-squared: 0.8978745164857953
MSE: 2.3645069433762367
MAE: 1.1919753277836762
```

[86]:
```python
X_train_final_const
```

[86]:

	const	TV	Radio
116	1.0	139.2	14.3
67	1.0	139.3	14.5

```
78     1.0    5.4   29.9
42     1.0  293.6   27.7
17     1.0  281.4   39.6
..     …     …     …
133    1.0  219.8   33.5
137    1.0  273.7   28.9
72     1.0   26.8   33.0
140    1.0   73.4   17.0
37     1.0   74.7   49.4

[140 rows x 3 columns]
```

[87]:
```
# Confirm the significant features in the final model
print("Significant features in the final model:", significant_features)
```

```
Significant features in the final model: ['TV', 'Radio']
```

[88]:
```
# Prepare new data based on only the significant features

# Creating the new input values dictionary with only significant features
new_data_values = {"TV": 100, "Radio": 50, "Newspaper": 70}
# Filter the dictionary to include only the significant features
new_data_final = pd.DataFrame(
    [{feature: new_data_values[feature] for feature in significant_features}]
)

# Add a constant to align with the model's intercept term
new_data_final_const = sm.add_constant(new_data_final, has_constant="add")
new_data_final_const
```

[88]:
```
   const   TV  Radio
0    1.0  100     50
```

[89]:
```
# Predict with a 90% confidence interval for the filtered input
predicted_sales = final_model.get_prediction(new_data_final_const)
ci_90 = predicted_sales.conf_int(alpha=0.1)  # 90% confidence level

print(f"90% Confidence Interval for sales prediction: {ci_90}")
```

```
90% Confidence Interval for sales prediction: [[14.71820447 15.72514686]]
```

[90]:
```
# Assumption Validation
# Residuals
residuals = final_model.resid
```

[91]:
```
# 1. Normality of Residuals (Jarque-Bera)
jb_stat, jb_pvalue, _, __ = jarque_bera(residuals)
print(f"Jarque-Bera Test: Statistic={jb_stat}, p-value={jb_pvalue}")
```

```
Jarque-Bera Test: Statistic=24.662861496030988, p-value=4.410904674141444e-06
```

[92]:
```python
# 2. Homoscedasticity (White's Test)
white_test = het_white(residuals, final_model.model.exog)
print(f"White's Test: Statistic={white_test[0]}, p-value={white_test[1]}")
```

```
White's Test: Statistic=11.353378151633706, p-value=0.04480664163871973
```

[93]:
```python
# 3. Autocorrelation (Durbin-Watson Test)
dw_stat = durbin_watson(residuals)
print(f"Durbin-Watson Statistic: {dw_stat}")
```

```
Durbin-Watson Statistic: 2.0375161916374998
```

[94]:
```python
# 4. Multicollinearity (Variance Inflation Factor - VIF)
vif_data = pd.DataFrame()
vif_data["feature"] = X_train_final.columns
vif_data["VIF"] = [
    variance_inflation_factor(X_train_final.values, i)
    for i in range(X_train_final.shape[1])
]
print(vif_data)
```

```
  feature       VIF
0      TV  2.111535
1   Radio  2.111535
```

[95]:
```python
# Interpret Results
if jb_pvalue > 0.1:
    print("The residuals are normally distributed (Jarque-Bera test).")
else:
    print("The residuals do not appear normally distributed (Jarque-Bera test).
 ↪")

if white_test[1] > 0.1:
    print("Homoscedasticity is present (White's test).")
else:
    print("Heteroscedasticity detected (White's test).")

if 1.5 < dw_stat < 2.5:
    print("No significant autocorrelation detected (Durbin-Watson test).")
else:
    print("Autocorrelation may be present (Durbin-Watson test).")

if vif_data["VIF"].max() < 10:
    print("No multicollinearity issues (VIF test).")
else:
    print("Multicollinearity detected (VIF test).")
```

```
The residuals do not appear normally distributed (Jarque-Bera test).
```

```
Heteroscedasticity detected (White's test).
No significant autocorrelation detected (Durbin-Watson test).
No multicollinearity issues (VIF test).
```