Projekt 02 - MSP

- Jakub Kratochvíl (xkrato67)

Věrohodnost

1. Zapište zvolenou parametrizaci Weibullova rozdělení, logaritmickou-věrohodnostní funkci pro zadaná data a její parciální derivace podle parametrů (shape, scale).
2. Pomocí scipy.optimize nalezněte maximálně věrohodné odhady parametrů weibullova rozdělení.
3. Pomocí věrohodnostního poměru otestujte hypotézu, že exponenciální rozdělení je postačujícím modelem zapsaných dat (Parametr tvaru = 1)
4. Podle výsledku ze 3) použijte výsledné rozdělení pravděpodobnosti (s maximálně věrohodnými odhady jako parametry) a nalezněte bodové odhady pro střední dobu zaměstnání v oboru a 10% percentil zaměstnání v oboru (za jakou dobu odejde do jiného odboru 10 % absolventů).
5. [dobrovolná část] zkuste nějak slovně charakterizovat/popsat fungování doby zaměstnání v oboru jako náhodné veličiny, dle Vašich výsledků a parametrů

1)

- Zvolená parametrizace: Zvolil jsem klasickou variantu weibullova rozdělení, které je definováno následovně:

$$f(\lambda, k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

- Logaritmická věrohodnostní funkce s cenzorovanými daty:

$$\ell(k, \lambda, \delta) = \sum_{i=1}^{n} \delta_i \left(\ln\left(\frac{k}{\lambda}\right) + (k-1)\ln\left(\frac{x_i}{\lambda}\right) - \left(\frac{x_i}{\lambda}\right)^k \right) + \\ + \sum_{i=1}^{n} (1-\delta_i) \left(-\left(\frac{x_i}{\lambda}\right)^k \right)$$

- Parciální derivace podle k a λ:

$$\frac{\partial \ell}{\partial k} = \sum_{i=1}^{n} \left[\delta_i \left(\frac{1}{k} + \ln\left(\frac{x_i}{\lambda}\right) - \left(\frac{x_i}{\lambda}\right)^k \ln\left(\frac{x_i}{\lambda}\right) \right) - (1-\delta_i)\left(\frac{x_i}{\lambda}\right)^k \ln\left(\frac{x_i}{\lambda}\right) \right]$$

$$\frac{\partial \ell}{\partial k} = \sum_{i=1}^{n} \delta_i \left(\frac{1}{k} + \ln\left(\frac{x_i}{\lambda}\right) \right) - \sum_{i=1}^{n} \left(\frac{x_i}{\lambda}\right)^k \ln\left(\frac{x_i}{\lambda}\right)$$

$$
\begin{aligned}
\frac{\partial \ell}{\partial \lambda} &= \sum_{i=1}^{n} \delta_i \left(-\frac{1}{\lambda} - \frac{k-1}{\lambda} + \frac{k x_i^k}{\lambda^{k+1}} \right) + \sum_{i=1}^{n} (1-\delta_i) \frac{k x_i^k}{\lambda^{k+1}} \\
\frac{\partial \ell}{\partial \lambda} &= \sum_{i=1}^{n} \delta_i \left(-\frac{1}{\lambda} - \frac{k-1}{\lambda} \right) + \frac{k x_i^k}{\lambda^{k+1}} \\
\frac{\partial \ell}{\partial \lambda} &= \sum_{i=1}^{n} \frac{k}{\lambda} \left(-\delta_i + \left(\frac{x_i}{\lambda} \right)^k \right)
\end{aligned}
$$

In [57]:
```python
import numpy as np
import pandas as pd
import scipy.stats as stats
from scipy.optimize import minimize
import scipy.special as special

# Načtení dat z Excelu
df = pd.read_excel('Data_2024.xlsx', sheet_name='Data_věrohodnost')

# Extrakce potřebných sloupců
times = df['doba práce v oboru [roky]'].values
censored = df['censored'].values.astype(bool)
```

In [58]:
```python
def weibull_log_likelihood(params, times, censored):
    k, lam = params
    t_per_l = times / lam
    pdf = k/lam * (t_per_l)**(k-1) * np.exp(-(t_per_l)**k)
    # Survival funkce (1 - CDF) pro cenzorovaná data
    survival_func = 1 - (1 - np.exp(-(t_per_l)**k))
    log_likelihood = np.sum(
        np.log(pdf) * ~censored + np.log(survival_func) * censored
    )
    return -log_likelihood

def get_mle(data_tuple):
    initial = [6.0, 7.0]
    result = minimize(
        weibull_log_likelihood,
        initial,
        args=data_tuple,
        method='Nelder-Mead'
    )
    # Výsledné parametry
    shape_param, scale_param = result.x
    print(f"Maximálně věrohodné odhady:")
    print(f"\tShape (k): {shape_param:.4f}")
    print(f"\tScale (λ): {scale_param:.4f}")
    return shape_param, scale_param

shape, scale = get_mle((times, censored))
```

```
Maximálně věrohodné odhady:
        Shape (k): 6.1728
        Scale (λ): 7.4295
```

2)

- MLE pro parametr Shape (k) vyšel: 6.1728

- MLE pro parametr Scale (λ) vyšel: 7.4295

In [59]:
```
#3. Pomocí věrohodnostního poměru otestujte hypotézu, že exponenciální rozdělení
#   postačujícím modelem zapsaných dat (Parametr tvaru = 1)

def likelihood_ratio_test(data_tuple, parameter_tuple):
    def constrained_likelihood(params):
        return weibull_log_likelihood(
            [1.0, params[0]],
            data_tuple[0],
            data_tuple[1],
        )

    constrained_result = minimize(
        constrained_likelihood,
        [parameter_tuple[0]],
        method='Nelder-Mead'
    )

    # Log-likelihood pro plný model
    full_log_likelihood = -weibull_log_likelihood(
        parameter_tuple,
        data_tuple[0],
        data_tuple[1]
    )
    constrained_log_likelihood = -constrained_result.fun

    # Likelihood ratio test statistic
    lr = 2 * (full_log_likelihood - constrained_log_likelihood)
    # Kritická hodnota pro 5% hladinu významnosti (1 stupeň volnosti)
    crit = stats.chi2.ppf(0.95, df=1)

    print("Likelihood Ratio Test:")
    print(f"\tTest statistic: {lr:.4f}")
    print(f"\tCritical value (5%): {crit:.4f}")
    print(f"\tZamítáme nulovou hypotézu: {'Ano' if lr > crit else 'Ne'}")

# Zavoláme metody pro vypsání hodnot
likelihood_ratio_test((times, censored), (shape, scale))
```

```
Likelihood Ratio Test:
        Test statistic: 592.3898
        Critical value (5%): 3.8415
        Zamítáme nulovou hypotézu: Ano
```

3)

H_0: Omezený exponenciální model je dostatečný

H_1: Plný model poskytuje lepší výsledek

Nulovou hypotézu zamítame, tedy s 95% pravděpodobností platí alternativní hypotéza - Plný model poskytuje lepší výsledek.

In [60]:
```
def weibull_point_estimates(parameters):
    shape = parameters[0]
    scale = parameters[1]
    # Střední doba zaměstnání
```

```
    mean_time = scale * special.gamma(1 + 1/shape)
    # 10% percentil
    percentile_10 = scale * (-np.log(0.9)) ** (1/shape)
    print("Charakteristiky:")
    print(f"\tStřední doba zaměstnání v oboru: {mean_time:.4f} let")
    print(f"\t10% percentil doby zaměstnání: {percentile_10:.4f} let")

weibull_point_estimates((shape,scale))
```

```
Charakteristiky:
        Střední doba zaměstnání v oboru: 6.9033 let
        10% percentil doby zaměstnání: 5.1598 let
```

4)

Střední doba zaměstnání v oboru vychází: 6.9033 let

10% percentil doby zaměstnání vychází: 5.1598 let

Regrese

1. Pomocí zpětné eliminace určete vhodný regresní model. Za výchozí „plný" model považujte plný kvadratický model (všechny interakce druhého řádu a všechny druhé mocniny, které dávají smysl).
   - Zapište rovnici Vašeho finálního modelu.
   - Diskutujte splnění předpokladů lineární regrese a základní regresní diagnostiky.
   - Pokud (až během regresního modelování) identifikujete některé „extrémně odlehlé hodnoty" můžete ty „nejodlehlejší" hodnoty, po alespoň krátkém zdůvodnění, vyřadit.
2. Pomocí vašeho výsledného modelu identifikujte, pro které nastavení parametrů má odezva nejproblematičtější (největší) hodnotu (použijte model, nikoli samotná pozorování).
3. Odhadněte hodnotu odezvy uživatele s Windows, při průměrném nastavení ostatních parametrů a vypočtěte konfidenční interval a predikční interval pro toto nastavení.
4. Na základě jakýchkoli vypočtených charakteristik argumentujte, zdali je Váš model „vhodný" pro další použit

1)

In [61]:
```
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Načtení dat z Excelu
df = pd.read_excel('Data_2024.xlsx', sheet_name='Data_regrese')
# Přejmenování, zejména kvůli formátu formule v smf.ols
df = df.rename(columns={
    'Ping [ms]'         : 'Ping',
    'InteractingPct'    : 'Interact',
    'ScrollingPct'      : 'Scroll',
```

```
})
# Přetypování OSType na categorical, jelikož máme jen 4 různé hodnoty
df['OSType'] = df['OSType'].astype('category')
# Z prvního pohledu na data vidíme, že procenta interakce a skrolování
# jsou sobě doplňkem, takže jeden z těchto sloupců pro regresní model
# nepotřebujeme. Odebereme třeba procenta interakce.
df = df.drop(columns=['Interact'])
df
```

Out[61]:

	OSType	ActiveUsers	Scroll	Ping
0	iOS	4113	0.1717	47
1	iOS	7549	0.6539	46
2	Windows	8855	0.7822	55
3	Android	8870	0.9206	56
4	MacOS	9559	0.2718	76
...	...	...	...	...
497	iOS	5315	0.8026	28
498	MacOS	1392	0.7627	24
499	iOS	6014	0.1888	54
500	Android	5118	0.7655	39
501	MacOS	2660	0.0610	55

502 rows × 4 columns

Nyní máme data připravená a můžeme vytvořit výchozí model. Jelikož nás v druhé otázce zajímá odezva (Ping), budeme v modelu vyjadřovat právě odezvu. Výchozí model bude definován následovně:

$$\begin{aligned} Ping = \beta_1 + \beta_2 * ActiveUsers + \beta_3 * Scroll + \beta_4 * Windows + \\ + \beta_5 * MacOS + \beta_6 * iOS + \beta_7 * (ActiveUsers * Scroll) + \\ + \beta_8 * (ActiveUsers * Windows) + \beta_9 * (ActiveUsers * MacOS) + \\ + \beta_{10} * (ActiveUsers * iOS) + \beta_{11} * ActiveUsers^2 + \beta_{12} * (Scroll * Windows) \\ + \beta_{13} * (Scroll * MacOS) + \beta_{14} * (Scroll * iOS) + \beta_{15} * Scroll^2 \end{aligned}$$

Nyní si model vytvoříme v kódu, kde to bude lehce ulehčené díky použití OSType jako kategoriálního typu, který nám zajistí rozepsání jednotlivých kategorií za nás, použitím C(OSType). Toto za nás dosadí dummies a také zajistí správnou multikolinearitu kategorických proměnných vynecháním jedné kategorie. Také ale zkontrolujeme multikolinearitu jednotlivých parametrů a případně provedeme standardizaci.

In [62]:

```
f = """
Ping ~ ActiveUsers + Scroll + C(OSType) + ActiveUsers:Scroll +
ActiveUsers:C(OSType) + I(ActiveUsers**2) + Scroll:C(OSType) + I(Scroll**2)
"""
model = smf.ols(formula=f, data=df)
```

```python
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.839
Method:                 Least Squares   F-statistic:                     187.9
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          5.18e-186
Time:                        13:32:46   Log-Likelihood:                -1598.4
No. Observations:                 502   AIC:                             3227.
Df Residuals:                     487   BIC:                             3290.
Df Model:                          14
Covariance Type:            nonrobust
==================================================================================
====================
                                       coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------------------
--------------------
Intercept                           33.5416      2.537     13.219      0.000
28.556      38.527
C(OSType)[T.MacOS]                   1.6452      2.313      0.711      0.477
-2.900       6.190
C(OSType)[T.Windows]                 8.2434      2.427      3.396      0.001
3.474      13.013
C(OSType)[T.iOS]                     0.2195      2.405      0.091      0.927
-4.506       4.945
ActiveUsers                          0.0069      0.001     11.780      0.000
0.006       0.008
ActiveUsers:C(OSType)[T.MacOS]       0.0014      0.000      4.536      0.000
0.001       0.002
ActiveUsers:C(OSType)[T.Windows]    -0.0008      0.000     -2.505      0.013
-0.001      -0.000
ActiveUsers:C(OSType)[T.iOS]        -0.0011      0.000     -3.369      0.001
-0.002      -0.000
Scroll                             -30.1546      4.528     -6.660      0.000
-39.051     -21.258
Scroll:C(OSType)[T.MacOS]            0.3566      2.530      0.141      0.888
-4.614       5.327
Scroll:C(OSType)[T.Windows]         -0.4260      2.721     -0.157      0.876
-5.771       4.919
Scroll:C(OSType)[T.iOS]             -0.2678      2.691     -0.100      0.921
-5.556       5.020
ActiveUsers:Scroll                   0.0031      0.000      8.532      0.000
0.002       0.004
I(ActiveUsers ** 2)               -4.17e-07    4.4e-08     -9.469      0.000   -
5.03e-07    -3.3e-07
I(Scroll ** 2)                      -3.7258      3.492     -1.067      0.287
-10.587       3.135
==============================================================================
Omnibus:                      228.442   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3152.488
Skew:                           1.603   Prob(JB):                         0.00
Kurtosis:                      14.851   Cond. No.                     1.06e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.06e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Vidíme, že dostáváme přímo upozornění na silnou multikolinearitu, což nám říká že bysme měli použít nějakou standardizaci. Tu tedy provedeme v následujícím kroku.

In [63]:
```
# VIF aktuálních dat
print("VIF dat před standardizací")
X = pd.DataFrame(model.exog, columns=model.exog_names)
vif = pd.Series([variance_inflation_factor(X.values, i)
                 for i in range(X.shape[1])],
                 index=X.columns)
vif_df = vif.to_frame()
vif_df.columns = ['VIF']
print(vif_df)

# VIF standardizovaného modelu pomocí z-value
print("\nVIF dat po standardizací")
dfs = df.copy()

# Uložíme si střední hodnoty a směrodatné hodnoty
# jelikož je budeme potřebovat později
au_mean = dfs['ActiveUsers'].mean()
au_std = dfs['ActiveUsers'].std()
dfs['ActiveUsers'] = (dfs['ActiveUsers'] - au_mean) / au_std

sc_mean = dfs['Scroll'].mean()
sc_std = dfs['Scroll'].std()
dfs['Scroll'] = (dfs['Scroll'] - sc_mean) / sc_std

model = smf.ols(formula=f, data=dfs)
results = model.fit()

X = pd.DataFrame(model.exog, columns=model.exog_names)
vif = pd.Series([variance_inflation_factor(X.values, i)
                 for i in range(X.shape[1])],
                 index=X.columns)
vif_df = vif.to_frame()
vif_df.columns = ['VIF']
print(vif_df)
```

```
VIF dat před standardizací
                                      VIF
Intercept                        91.854488
C(OSType)[T.MacOS]               15.217934
C(OSType)[T.Windows]             16.447197
C(OSType)[T.iOS]                 14.840501
ActiveUsers                      32.087036
ActiveUsers:C(OSType)[T.MacOS]   10.454073
ActiveUsers:C(OSType)[T.Windows] 10.189613
ActiveUsers:C(OSType)[T.iOS]      9.061267
Scroll                           25.577021
Scroll:C(OSType)[T.MacOS]         6.489601
Scroll:C(OSType)[T.Windows]       7.671741
Scroll:C(OSType)[T.iOS]           7.692924
ActiveUsers:Scroll                8.705628
I(ActiveUsers ** 2)              22.499134
I(Scroll ** 2)                   16.422293

VIF dat po standardizací
                                     VIF
Intercept                        6.781856
C(OSType)[T.MacOS]               1.654677
C(OSType)[T.Windows]             1.630480
C(OSType)[T.iOS]                 1.606613
ActiveUsers                      4.918200
ActiveUsers:C(OSType)[T.MacOS]   2.335540
ActiveUsers:C(OSType)[T.Windows] 2.383411
ActiveUsers:C(OSType)[T.iOS]     2.247663
Scroll                           4.737489
Scroll:C(OSType)[T.MacOS]        2.503341
Scroll:C(OSType)[T.Windows]      2.092440
Scroll:C(OSType)[T.iOS]          2.198356
ActiveUsers:Scroll               1.041103
I(ActiveUsers ** 2)              1.019635
I(Scroll ** 2)                   1.032984
```

Po standardizaci sloupců ActiveUsers a Scroll jsou aktuální hodnoty VIF menší než hodnota 10, což už bereme jako přijatelné hodnoty míry multikolinearity.

Při následném dosazování hodnot nesmíme ale zapomenout nové hodnoty buď standardizovat anebo navrátit podel do původního tvaru.

Jelikož máme připravený výchozí model, můžeme konečně provést zpětnou eliminaci, která funguje následovně:

1. Začneme se všemi přomennými v modelu
2. Odebereme proměnnou s největší p-hodnotou
3. Použijeme aktualizováný model a vracíme se do bodu 2 až do nějaké zastavovací podmínky, např. kdy všechny proměnné mají p-hodnotu pod nějakým prahem

Tím, že odstraníme hodnoty s největšími p-hodnotami tak odebereme statisticky nezávislé proměnné.

Začal jsem tedy s implementací zpětné eliminace, ovšem po napsání jakéhosi prototypu mi došlo, že nedokážu odstanit z řetezce formule specifickou kategorii, jelikož ty se automaticky rozbalují z C(OSType).

A jelikož se mi už něchtělo upravovat dataframe a dělat si vlastní kategorie našel jsem workaround a to takový, že přímo upravuji matici, kde odstraňuji nežádoucí kategorie nebo interakce.

Zpětnou eliminaci implementuje následující blok kódu:

In [64]:
```
import patsy as pat
# Funkce pro úpravu modelu pomocí modifikace matice
def remove_from_matrix(formula, dataframe, remove_str, columns_to_drop_prev):
    y, X = pat.dmatrices(formula, data=dataframe, return_type='dataframe')

    if isinstance(remove_str, tuple):
        new_columns_to_drop = [
            col for col in X.columns if all(r in col for r in remove_str)
        ]
    else:
        new_columns_to_drop = [
            col for col in X.columns if remove_str in col
        ]

    columns_to_drop = list(set(columns_to_drop_prev + new_columns_to_drop))
    X_modified = X.drop(columns=columns_to_drop)
    model = sm.OLS(y, X_modified)
    return model, columns_to_drop


# Pomocná funkce pro získání proměnných v intersekci
def parse_interaction(s):
    return s if ":" not in s else s.split(':')[0], s.split('[T.')[-1].strip(']')


# Výchozí standardizovaný model je uložen v proměnné model a fit() v result
p_values = results.pvalues
max_value = p_values[p_values.idxmax()]
s1,s2 = parse_interaction(p_values.idxmax())
print("Variable with the highest p-value is: ", s1+":"+s2, max_value)
new_cols = []

# Loop který provádí zpětnou eliminaci a vypisuje odstraňované proměnné
while max_value > 0.05:
    inter = parse_interaction(p_values.idxmax())
    model, new_cols = remove_from_matrix(f, dfs, inter, new_cols)
    results = model.fit()
    p_values = results.pvalues
    max_value = p_values[p_values.idxmax()]
    s1,s2 = parse_interaction(p_values.idxmax())
    print("Variable with the highest p-value is: ", s1+":"+s2, max_value)

print("\nFinal model:")
results_final = results
print(results_final.summary())
```

```
Variable with the highest p-value is:  Scroll:iOS 0.920766522138589
Variable with the highest p-value is:  Scroll:Windows 0.9023977373454168
Variable with the highest p-value is:  Scroll:MacOS 0.760220507487699
Variable with the highest p-value is:  I(Scroll ** 2):I(Scroll ** 2) 0.2775322305
7943377
Variable with the highest p-value is:  ActiveUsers:Windows 0.014940511032521005

Final model:
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     264.4
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          1.69e-190
Time:                        13:32:46   Log-Likelihood:                -1599.1
No. Observations:                 502   AIC:                             3220.
Df Residuals:                     491   BIC:                             3267.
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
===================
                                       coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------------
Intercept                           51.0013      0.620     82.217      0.000
49.782      52.220
C(OSType)[T.MacOS]                   9.4142      0.757     12.438      0.000
7.927      10.901
C(OSType)[T.Windows]                 3.8746      0.761      5.090      0.000
2.379       5.370
C(OSType)[T.iOS]                    -5.7581      0.787     -7.319      0.000
-7.304      -4.212
ActiveUsers                         10.0146      0.585     17.124      0.000
8.866      11.164
ActiveUsers:C(OSType)[T.MacOS]       3.6179      0.781      4.633      0.000
2.084       5.152
ActiveUsers:C(OSType)[T.Windows]    -1.8783      0.769     -2.442      0.015
-3.389      -0.367
ActiveUsers:C(OSType)[T.iOS]        -2.6316      0.796     -3.306      0.001
-4.196      -1.068
Scroll                              -5.0399      0.266    -18.931      0.000
-5.563      -4.517
ActiveUsers:Scroll                   2.3325      0.269      8.662      0.000
1.803       2.862
I(ActiveUsers ** 2)                 -2.7035      0.285     -9.480      0.000
-3.264      -2.143
==============================================================================
Omnibus:                      230.750   Durbin-Watson:                   1.928
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3263.977
Skew:                           1.617   Prob(JB):                         0.00
Kurtosis:                      15.066   Cond. No.                         8.06
==============================================================================

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

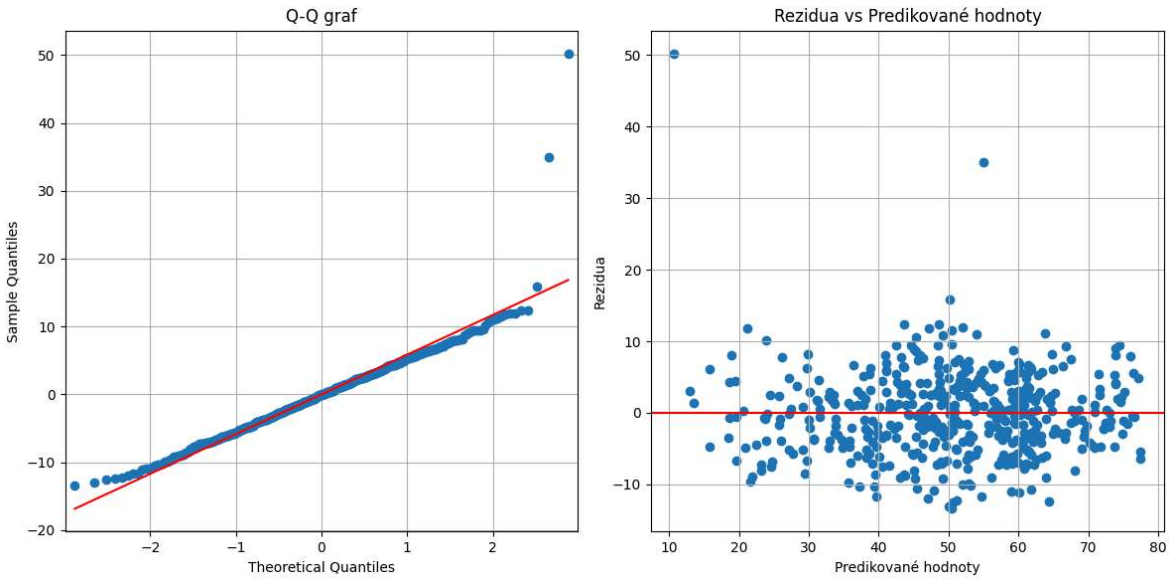
Nalezli jsme tedy náš výsledný model pomocí zpětné eliminace. Rovnice modelu tedy vypadá následovně:

$$
\begin{aligned}
Ping = \beta_1 + \beta_2 * ActiveUsers + \beta_3 * Scroll + \beta_4 * Windows + \\
+ \beta_5 * MacOS + \beta_6 * iOS \beta_7 * (ActiveUsers * Scroll) + \\
+ \beta_8 * (ActiveUsers * Windows) + \beta_9 * (ActiveUsers * MacOS) + \\
+ \beta_{10} * (ActiveUsers * iOS) + \beta_{11} * ActiveUsers^2
\end{aligned}
$$

Dalším úkolem je odstranit odlehlé hodnoty. Ideálně bychom to měli provést už před standardizací, ale vzhledem k tomu, že jsme doposud nevěděli, jak bude náš model vypadat, je stále dobré to udělat alespoň nyní. Data si zobrazíme do grafů.

In [65]:
```
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
from statsmodels.graphics.tsaplots import plot_acf

fig, ax = plt.subplots(1, 2, figsize=(12, 6))
qqplot(results_final.resid, line='s', ax=ax[0])
ax[0].set_title('Q-Q graf')
ax[0].grid(True)
ax[1].scatter(results_final.fittedvalues, results_final.resid)
ax[1].axhline(y=0, color='r', linestyle='-')
ax[1].grid(True)
ax[1].set_xlabel('Predikované hodnoty')
ax[1].set_ylabel('Rezidua')
ax[1].set_title('Rezidua vs Predikované hodnoty')
plt.tight_layout()
plt.show()
```



Z grafů můžeme vidět, že mám dvě odlehlé hodnoty, které se velice odchylují od predikovaných (normálních) hodnot. Tyto dvě hodnoty můžeme z modelu dostat pomocí Cookovu vzdáleností a nastavení vhodného prahu, tak abychom smazal pouze tyto dvě hodnoty.

Ovšem je vždycky dobré mít nějaký hlubší důvod pro odebrání dat než pouze test nebo odhad. V tomto případě bych to odůvodnil tak, že máme docela velký počet pozorování a vychýlení je opravdu značné, proto si myslím že to je v tomto případě validní.

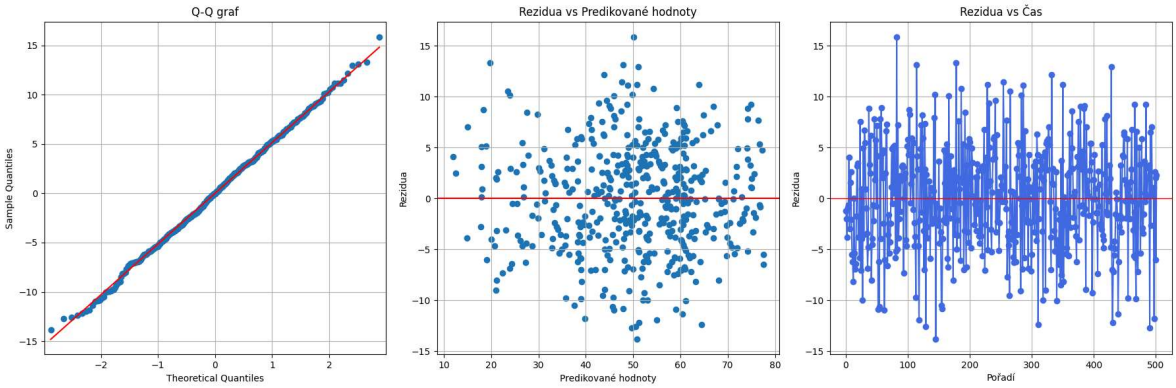
Nakonec natrénujeme model bez těchto odlehlých hodnot.

In [66]:
```
# Výpočet odlehlých hodnot z Cookovi vzdálenosti
cook = results.get_influence().cooks_distance[0]
threshold = 15 / len(results.model.endog)
outliers = np.where(cook > threshold)[0]
print(f"Indexy odlehlých hodnot: {outliers}")
dfs_clean = dfs.drop(index=outliers)

# Přepočítání modelu
f_final = """
Ping ~ ActiveUsers + Scroll + C(OSType) + ActiveUsers:Scroll +
ActiveUsers:C(OSType) + I(ActiveUsers**2)
"""
model_clean = smf.ols(formula=f_final, data=dfs_clean)
results_final_clean = model_clean.fit()

fig, ax = plt.subplots(1, 3, figsize=(18, 6))
qqplot(results_final_clean.resid, line='s', ax=ax[0])
# Q-Q graf
ax[0].set_title('Q-Q graf')
ax[0].grid(True)
# Rezidua vs. Predikce graf
ax[1].scatter(results_final_clean.fittedvalues, results_final_clean.resid)
ax[1].axhline(y=0, color='r', linestyle='-')
ax[1].grid(True)
ax[1].set_xlabel('Predikované hodnoty')
ax[1].set_ylabel('Rezidua')
ax[1].set_title('Rezidua vs Predikované hodnoty')
# Rezidua vs. Čas graf
ax[2].plot(
    results_final_clean.resid,
    marker='o',
    linestyle='-',
    color='royalblue'
)
ax[2].axhline(y=0, color='r', linestyle='-', linewidth=1)
ax[2].grid(True)
ax[2].set_title('Rezidua vs Čas')
ax[2].set_xlabel('Pořadí')
ax[2].set_ylabel('Rezidua')
plt.tight_layout()
plt.show()
```

```
Indexy odlehlých hodnot: [255 476]
```



Vypíšeme si shrnutí modelu a VIF pro finální očištění model.

In [67]:
```
print(results_final_clean.summary())
model = smf.ols(formula=f_final, data=dfs_clean)
```

```python
results = model.fit()
X = pd.DataFrame(model.exog, columns=model.exog_names)
vif = pd.Series([variance_inflation_factor(X.values, i)
                 for i in range(X.shape[1])],
                 index=X.columns)
vif_df = vif.to_frame()
vif_df.columns = ['VIF']
print(vif_df)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.875
Method:                 Least Squares   F-statistic:                     349.9
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          1.28e-215
Time:                        13:32:46   Log-Likelihood:                -1528.7
No. Observations:                 500   AIC:                             3079.
Df Residuals:                     489   BIC:                             3126.
Df Model:                          10
Covariance Type:            nonrobust
==================================================================================
====================
                                       coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------------------
--------------------
Intercept                           51.2936      0.547     93.743      0.000
50.219      52.369
C(OSType)[T.MacOS]                   9.0073      0.667     13.501      0.000
7.696      10.318
C(OSType)[T.Windows]                 3.6657      0.671      5.463      0.000
2.347       4.984
C(OSType)[T.iOS]                    -5.7223      0.692     -8.267      0.000
-7.082      -4.362
ActiveUsers                         10.0457      0.515     19.522      0.000
9.035      11.057
ActiveUsers:C(OSType)[T.MacOS]       4.4373      0.692      6.415      0.000
3.078       5.796
ActiveUsers:C(OSType)[T.Windows]    -1.9127      0.677     -2.827      0.005
-3.242      -0.583
ActiveUsers:C(OSType)[T.iOS]        -2.7423      0.700     -3.915      0.000
-4.119      -1.366
Scroll                              -5.1410      0.234    -21.928      0.000
-5.602      -4.680
ActiveUsers:Scroll                   2.5596      0.238     10.752      0.000
2.092       3.027
I(ActiveUsers ** 2)                 -3.0081      0.254    -11.832      0.000
-3.508      -2.509
==============================================================================
Omnibus:                        0.661   Durbin-Watson:                   1.990
Prob(Omnibus):                  0.719   Jarque-Bera (JB):                0.750
Skew:                           0.014   Prob(JB):                        0.687
Kurtosis:                       2.812   Cond. No.                         8.01
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
                                       VIF
Intercept                         5.526351
C(OSType)[T.MacOS]                1.634335
C(OSType)[T.Windows]              1.622745
C(OSType)[T.iOS]                  1.594566
ActiveUsers                       4.854331
ActiveUsers:C(OSType)[T.MacOS]    2.277265
ActiveUsers:C(OSType)[T.Windows]  2.359393
ActiveUsers:C(OSType)[T.iOS]      2.233880
Scroll                            1.014852
```

```
ActiveUsers:Scroll                1.021173
I(ActiveUsers ** 2)               1.016205
```

Finální rovnice po dosazení:

$$
\begin{aligned}
Ping = 51.2936 + 10.0457 * ActiveUsers - 5.1410 * Scroll + 3.6657 * Windows + \\
+ 9.0073 * MacOS - 5.7223 * iOS + 2.5596 * (ActiveUsers * Scroll) + \\
- 1.9127 * (ActiveUsers * Windows) + 4.4373 * (ActiveUsers * MacOS) - \\
- 2.7423 * (ActiveUsers * iOS) - 3.0081 * ActiveUsers^2
\end{aligned}
$$

Splnění předpokladů regrese:

- Normalita - Víme že data normální jsou podle Q-Q grafu o kousek výše, jelikož leží přímo na přímce.
- Homoskedasticita - Tu jsme schopni ověřit v grafu Reziduí a Predikcí, kde vidíme že data mají v celku různé rozptyly a nemají žádnou očividnou závislost, což je správně.
- Autokorelace - Z grafu Reziduí a Pořadí jsme schopni ověřit, že data nejsou nijak závislá na pořadí nebo čase.
- Multikolinearita - Po standardizaci a odstranění odlehlých hodnot mají všechny parametry míru multikolinearity změřenou pomocí VIF menší jak 10, což bereme jako přijatelné.

Můžeme tedy usoudit, že základní regresní předpoklady jsou splněny. Splnění těchto předpokladů nám umožňuje považovat výsledky regresní analýzy za spolehlivé a interpretovatelné. Regrese tak poskytuje validní odhad vztahů mezi proměnnými.

2)

Nejprve si vygenerujeme si predikovaná data, najdeme maximální hodnotu a tu použijeme na nalezení hodnot konfigurace.

Zároveň nesmíme zapomenout "destanardardizovat" sloupec ActiveUsers a Scroll na puvodní podobnu před standardizací.

In [68]:
```
predicted = results_final_clean.predict(dfs_clean)
params = dfs_clean.iloc[predicted.idxmax()].copy()
max_value = predicted[predicted.idxmax()]

# Destandardizace
params['ActiveUsers'] = params['ActiveUsers'] * au_std + au_mean
params['Scroll'] = params['Scroll'] * sc_std + sc_mean
print("Max predicted ping: ", max_value)
print("\nProblematic parameters: ")
print(params)
```

```
Max predicted ping:  77.50462838943051

Problematic parameters:
OSType          MacOS
ActiveUsers    9657.0
Scroll          0.027
Ping               72
Name: 10, dtype: object
```

Nejproblematičtější odezvu má výsledný model pro OSType: MacOS, ActiveUsers: 9657 a Scroll: 0.027.

Maximální predikovaná hodnota odezvy je: 77.504628 ms

3)

In [69]:
```
# Vytvořený průměrné Windows konfigurace
windows_average = pd.DataFrame({
    'ActiveUsers':  [dfs_clean['ActiveUsers'].mean()],
    'Scroll':       [dfs_clean['Scroll'].mean()],
    'OSType':       ['Windows'],
})
win_cpy = windows_average.copy()
win_cpy['ActiveUsers'] = win_cpy['ActiveUsers'] * au_std + au_mean
win_cpy['Scroll'] = win_cpy['Scroll'] * sc_std + sc_mean
print("Average values for Windows:")
print(win_cpy)

# Pořebujeme použít get_prediction abychom měli summary_frame()
predicted = results_final_clean.get_prediction(windows_average)
sum = predicted.summary_frame(alpha=0.05)
print("\nInterval thresholds:")
print(sum[[
    'mean',
    'obs_ci_lower',
    'obs_ci_upper',
    'mean_ci_lower',
    'mean_ci_upper'
]])
```

```
Average values for Windows:
   ActiveUsers    Scroll   OSType
0     5496.442  0.510837  Windows

Interval thresholds:
        mean  obs_ci_lower  obs_ci_upper  mean_ci_lower  mean_ci_upper
0  55.002689     44.724693     65.280685      53.972145      56.033232
```

Průměrné nastavení pro Windows je ActiveUsers: 45496.442, Scroll: 0.510837 a samozřejmě OSType: Windows.

Výsledné intervaly jsou tedy následující:

- Interval spolehlivosti 95%: <53.9721; 56.0332>
- Predikční interval 95%: <44.7247; 65.2807>

Odhad odezvy uživatele Windows s průměrnými parametry: 55.0027 ms

4)

Nakonec is opět zobrazíme přehled o modelu.

In [70]:
```
print(results_final_clean.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.875
Method:                 Least Squares   F-statistic:                     349.9
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          1.28e-215
Time:                        13:32:47   Log-Likelihood:                -1528.7
No. Observations:                 500   AIC:                             3079.
Df Residuals:                     489   BIC:                             3126.
Df Model:                          10
Covariance Type:            nonrobust
==================================================================================
===================
                                      coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------------------
-------------------
Intercept                          51.2936      0.547     93.743      0.000
50.219      52.369
C(OSType)[T.MacOS]                  9.0073      0.667     13.501      0.000
7.696      10.318
C(OSType)[T.Windows]                3.6657      0.671      5.463      0.000
2.347       4.984
C(OSType)[T.iOS]                   -5.7223      0.692     -8.267      0.000
-7.082      -4.362
ActiveUsers                        10.0457      0.515     19.522      0.000
9.035      11.057
ActiveUsers:C(OSType)[T.MacOS]      4.4373      0.692      6.415      0.000
3.078       5.796
ActiveUsers:C(OSType)[T.Windows]   -1.9127      0.677     -2.827      0.005
-3.242      -0.583
ActiveUsers:C(OSType)[T.iOS]       -2.7423      0.700     -3.915      0.000
-4.119      -1.366
Scroll                             -5.1410      0.234    -21.928      0.000
-5.602      -4.680
ActiveUsers:Scroll                  2.5596      0.238     10.752      0.000
2.092       3.027
I(ActiveUsers ** 2)                -3.0081      0.254    -11.832      0.000
-3.508      -2.509
==============================================================================
Omnibus:                        0.661   Durbin-Watson:                   1.990
Prob(Omnibus):                  0.719   Jarque-Bera (JB):                0.750
Skew:                           0.014   Prob(JB):                        0.687
Kurtosis:                       2.812   Cond. No.                         8.01
==============================================================================

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

Model je statisticky silný a hodnota R-squared 0,877 ukazuje, že model dokáže vysvětlit 87,7 % změn v hodnotách Ping na základě proměnných zahrnutých v modelu.

Dále všechny prediktory jsou statisticky významné jelikož jejich p-hodnota je menší než 0.05.

F-statistika 349.9 potvrzuje, že model jako celek je relevantní. Analýza reziduí žádné závažné porušení předpokladů OLS (rezidua jsou normálně rozložena, bez autokorelace).

Nízké číslo podmínky 8.01 naznačuje, že multikolinearita není problém, což jsem již kontrolvali pomocí VIF pár kroků zpět.

Celkově model splňuje statistické požadavky a je vhodný k dalšímu použití, ale musíme brát v potaz, že větší počet pozorování by modelu určitě prospěl, kvůli odstraněným odlehlým hodnotám.