MSP projekt 2 - Ondřej Lukášek (xlukas15)

Začnu tím, že si naimportuji potřebné knihovny, se kterými budu v projektu pracovat.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.optimize as opt
import scipy.stats as stats
import scipy.special as special
import statsmodels.formula.api as smf

from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.gofplots import qqplot
```

Úkol 1 - Věrohodnost

Nejprve si načtu list (sheet), který obsahuje data, se kterými budu pracovat.

Předpřipravím si parametry tak, abych je později mohl rovnou použít.

In [2]:
```
file_path = 'Data_2024.xlsx'
sheet_name = 'Data_věrohodnost'

data = pd.read_excel(file_path, sheet_name=sheet_name)
times = data['doba práce v oboru [roky]'].dropna().to_numpy()
censored = data['censored'].to_numpy()

initial_params = (1.5, 5.0)
```

1.1

Zapište zvolenou parametrizaci Weibullova rozdělení, logaritmickou-věrohodnostní funkci pro zadaná data a její parciální derivace podle parametrů (shape, scale).

Funkce hustoty pravděpodobnosti Weibullova rozdělení vypadá takto (vizte stránku na Wikipedii):

$$f(x;k,\lambda) = \frac{k}{\lambda} \cdot \left(\frac{x}{\lambda}\right)^{k-1} \cdot e^{-(\frac{x}{\lambda})^k}, \quad x \geq 0$$

Log-likelihood potom je:

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

Nicméně tato rovnice obsahuje pouze část pro necenzorovaná data. Musím ještě přidat **survival funkci pro cenzorovaná data**, která vypadá následovně:

$$S(x;k,\lambda) = 1 - F(x;k,\lambda) = e^{-(\frac{x}{\lambda})^k}$$

Výsledný log-likelihood tedy bude vypadat následovně:

$$\ell(x;k,\lambda) = \sum_{i=1}^{n} \left[(1-c_i) \cdot \left(\ln(k) + (k-1) \cdot \ln(x_i) - k \cdot \ln(\lambda) - \left(\frac{x_i}{\lambda}\right)^k\right) + c_i \cdot \left(-\left(\frac{x_i}{\lambda}\right)^k\right)\right]$$

Přičemž c_i značí cenzorovanost daného data (`True/False`).

Parciální derivace pak jsou:

Podle k:

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

Podle λ:

$$\frac{\partial \ell}{\partial \lambda} = \sum_{i=1}^{n} \left[(1-c_i) \cdot \left(\frac{-k}{\lambda} - k \cdot \left(\frac{x_i}{\lambda}\right)^{k-1} \cdot \frac{1}{-\lambda^2}\right) + c_i \cdot \left(-k \cdot \left(\frac{x_i}{\lambda}\right)^{k-1} \cdot \frac{1}{-\lambda^2}\right)\right]$$

In [3]:
```python
def weibull_log_likelihood(params, times, censored):
    k, lam = params

    # logaritmizace pro necenzurovana data
    log_uncensored = np.log(k) + (k - 1) * np.log(times) - k * np.log(lam) - (times / lam)**k
    # logaritmizace pro cenzurovana data
    log_censored = - (times / lam)**k

    # spojeni logaritmu pro necenzurovana a cenzurovana data
    likelihood = (1 - censored) * log_uncensored + censored * log_censored

    return likelihood.sum()


def weibull_log_likelihood_derivatives(params, times, censored):
    k, lam = params

    # predvypocitam si veci, co se opakuji pro obe casti
    der_k_util_frac = np.log(times / lam) * (times / lam)**k
    der_lam_util_frac = k * (times / lam)**(k - 1) * (1 / - lam**2)

    # cast pro necezurovana
    der_k_uncensored = (1 / k) + np.log(times) - np.log(lam) - der_k_util_frac
    der_lam_uncensored = (-k / lam) - der_lam_util_frac

    # cast pro cenzurovana
    der_k_censored = -der_k_util_frac
    der_lam_censored = - der_lam_util_frac

    # vypocet derivaci
    der_k = (1 - censored) * der_k_uncensored + censored * der_k_censored
    der_lam = (1 - censored) * der_lam_uncensored + censored * der_lam_censored

    # vratim sumy derivaci
    return np.array([np.sum(der_k), np.sum(der_lam)])


log_likelihood = weibull_log_likelihood(initial_params, times, censored)
derivatives = weibull_log_likelihood_derivatives(initial_params, times, censored)

print(f'Log-likelihood: {round(log_likelihood, 4)}\n')
print(f'Derivatives:')
print(f'\t Shape (k): {round(derivatives[0], 4)}')
print(f'\t Scale (lambda): {round(derivatives[1], 4)}')
```

```
Log-likelihood: -740.8977

Derivatives:
         Shape (k): 64.242
         Scale (lambda): -47.9158
```

1.2

Pomocí `scipy.optimize` nalezněte maximálně věrohodné odhady parametrů weibullova rozdělení.

Scipy bohužel disponuje jenom metodou `minimize()`, takže si budu muset funkci pro log likelihood "otočit" (dát před ni mínus), a hledat její minimum, což pro "neotočenou" funkci bude maximum, které chci najít.

In [4]:
```python
# optimize ma jenom minimize, proto musim dat minus
def neg_weibull_log_likelihood(params, times, censored):
    return -weibull_log_likelihood(params, times, censored)


# vypocitam optimalni parametry
result = opt.minimize(
    fun=neg_weibull_log_likelihood,
    x0=initial_params,
    args=(times, censored),
    method='L-BFGS-B',
    bounds=[(0.1, None), (0.1, None)]
)

optimal_shape, optimal_scale = result.x

print('=== Maximal likelihood estimation ===')
print(f'Shape (k): {round(optimal_shape, 4)}')
print(f'Scale (lambda): {round(optimal_scale, 4)}')
```

```
=== Maximal likelihood estimation ===
Shape (k): 6.1728
Scale (lambda): 7.4295
```

1.3

Pomocí věrohodnostního poměru otestujte hypotézu, že exponenciální rozdělení je postačujícím modelem zapsaných dat (Parametr tvaru = 1).

By default je distribuční funkce (pro $x \geq 0$) exponenciálního rozdělení:

$$f(x;\lambda) = \lambda \cdot e^{-\lambda x}$$

Logaritmus potom bude:

$$\ell(x;\lambda) = \ln(\lambda) - \lambda \cdot x$$

Log-likelihood potom je:

$$\ell(x;\lambda) = \sum_{i=1}^{n} \left[\ln(\lambda) - \lambda x_i \right]$$

In [5]:
```python
# napisu si likelihood exponencialniho rozlozeni
def exponential_log_likelihood(lam, times, censored):
    log_f = np.log(lam) - lam * times
    log_sf = -lam * times

    likelihood = (1 - censored) * log_f + censored * log_sf
    return np.sum(likelihood)


# maximalni likelihood pro exponencialni rozlozeni
result_exp = opt.minimize(
    fun=lambda lam: -exponential_log_likelihood(lam[0], times, censored),
    x0=[1.0],
    bounds=[(0.1, None)],
    method='L-BFGS-B'
)

lambda_exp = result_exp.x[0]
log_likelihood_exp = -result_exp.fun

k_weibull, lambda_weibull = result.x
result_weibull = -result.fun

# VYPOCET VEROHODNOSTNIHO POMERU
LR = 2 * (result_weibull - log_likelihood_exp)

alpha = 0.05
critical_value = stats.chi2.ppf(1 - alpha, df=1)

print(f'Likelihood ratio: {round(LR, 4)}')
print(f'Critical value (alpha = {alpha}): {round(critical_value, 4)}')

if LR > critical_value:
    print('Zamítáme nulovou hypotézu. Exponenciální rozdělení není postačující.')
else:
    print('Nulovou hypotézu nezamítáme. Exponenciální rozdělení je postačující.')
```

```
Likelihood ratio: 592.3898
Critical value (alpha = 0.05): 3.8415
Zamítáme nulovou hypotézu. Exponenciální rozdělení není postačující.
```

1.4

Podle výsledku z předchozího bodu 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 oboru 10% absolventů).

Jelikož v předchozím bodu na základě testu vyšlo, že exponenciální rozdělení není postačující, budu dále pokračovat s Weibullovým rozdělením.

U Weibullova rozdělení se střední hodnota vypočítá následovně (z Wikipedie):

$$Mean = \lambda\Gamma(1 + \frac{1}{k})$$

A kvantily se počítají pomocí vzorce:

$$Q(p) = \lambda(-\ln(1-p))^{\frac{1}{k}} \implies Q(0.1) = \lambda(-\ln(0.9))^{\frac{1}{k}}$$

In [6]:
```python
# stredni hodnota weiibullova rozlozeni
def weibull_mean(k, lam):
    return lam * special.gamma(1 + 1 / k)


# kvantil weiibullova rozlozeni
def weibull_quantile(p, k, lam):
    return lam * (- np.log(1 - p))**(1 / k)


mean = weibull_mean(k_weibull, lambda_weibull)
quantile = weibull_quantile(0.1, k_weibull, lambda_weibull)

print('=== Weibull distribution ===')
print(f'Shape (k): {round(k_weibull, 4)}')
print(f'Scale (lambda): {round(lambda_weibull, 4)}\n')

print('=== Mean, Quantile ===')
print(f'Mean: {round(mean, 4)}')
print(f'10% quantile: {round(quantile, 4)}')
```

```
=== Weibull distribution ===
Shape (k): 6.1728
Scale (lambda): 7.4295

=== Mean, Quantile ===
Mean: 6.9032
10% quantile: 5.1598
```

1.5 (dobrovolný)

Zkuste nějak slovně charakterizovat/popsat fungování doby zaměstnání v oboru jako náhodné veličiny, dle vašich výsledků a parametrů.

Doba změstnání v oboru se řídí Weibullovým rozdělením o parametrech:

- $k \approx 6.1728$
- $\lambda \approx 7.4295$

Poměrně vysoké k znamená, že že pravděpodobnost, že absolvent opustí obor, výrazně roste s časem. Zároveň to také vypadá, že všichni opouštějí obor po velmi podobné době.

Průměrná doba setrvání v oboru je přibližně `6.9` let, přičemž `10%` absolventů obor opustí během zhruba `5.2` let.

Úkol 2 - Regrese

Opět začnu tím, že si nejprve načtu data z Excelu. Pro jistotu si přetypuji hodnoty.

In [7]:
```
file_path = 'Data_2024.xlsx'
sheet_name = 'Data_regrese'

df = pd.read_excel(file_path, sheet_name=sheet_name)

df['OSType'] = df['OSType'].astype('category')

columns_to_numeric = ['ActiveUsers', 'InteractingPct', 'ScrollingPct', 'Ping [ms]']
df[columns_to_numeric] = df[columns_to_numeric].apply(pd.to_numeric, errors='coerce')

# prejmenuji si sloupcek Ping [ms] na Ping
df.rename(columns={'Ping [ms]': 'Ping'}, inplace=True)

df.head()
```

Out[7]:

	OSType	ActiveUsers	InteractingPct	ScrollingPct	Ping
0	iOS	4113	0.8283	0.1717	47
1	iOS	7549	0.3461	0.6539	46
2	Windows	8855	0.2178	0.7822	55
3	Android	8870	0.0794	0.9206	56
4	MacOS	9559	0.7282	0.2718	76

Nyní si znormalizuji hodnoty pro sloupec aktivních uživatelů.

In [8]:
```
# udelam si kopii, abych mohl potom denormalizovat
df_original = df.copy()

# provedu normalizaci na <-1, 1>
df['ActiveUsers'] = 2 * (df['ActiveUsers'] - (df['ActiveUsers'].max() + df['ActiveUsers'].min()) / 2) / \
                    (df['ActiveUsers'].max() - df['ActiveUsers'].min())

df.head()
```

Out[8]:

	OSType	ActiveUsers	InteractingPct	ScrollingPct	Ping
0	iOS	-0.191837	0.8283	0.1717	47
1	iOS	0.509388	0.3461	0.6539	46
2	Windows	0.775918	0.2178	0.7822	55
3	Android	0.778980	0.0794	0.9206	56
4	MacOS	0.919592	0.7282	0.2718	76

Připravím si i funkci pro denormalizaci, protože bych ji mohl potřebovat (spoiler: budu ji potřebovat).

In [9]:
```
def denormalize_value(value):
    global df_original

    min_active = df_original['ActiveUsers'].min()
    max_active = df_original['ActiveUsers'].max()

    denormalized_value = value * (max_active - min_active) / 2 + (max_active + min_active) / 2

    return denormalized_value
```

Vytvořím si dummy sloupečky pro operační systémy tak, že každý OS bude mít svůj sloupeček (kromě Androidu, ten je reference). Hodnoty ve sloupečcích OS mohou nabývat hodnoty `True` nebo `False`.

In [10]:
```
# prevod na dummy promenne - Android dropneme, protoze bude refencni kategorii
dummy_ostype = pd.get_dummies(df['OSType'], drop_first=True)

# sloucime je s puvodnim dataframem a dropneme puvodni sloupec, protoze neni potreba
df = pd.concat([df, dummy_ostype], axis=1)
df = df.drop(columns=['OSType'])

print(df.head())
```

```
   ActiveUsers  InteractingPct  ScrollingPct  Ping  MacOS  Windows    iOS
0    -0.191837          0.8283        0.1717    47  False    False   True
1     0.509388          0.3461        0.6539    46  False    False   True
2     0.775918          0.2178        0.7822    55  False     True  False
3     0.778980          0.0794        0.9206    56  False    False  False
4     0.919592          0.7282        0.2718    76   True    False  False
```

2.1

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 přepokladů lineární regrese a základní regresní diagnostiky.
- Pokud (až během regresního modelování) identifikujete některé "nejodlehlejší" hodnoty, po alespoň krátkém zdůvodnění, vyřadit.

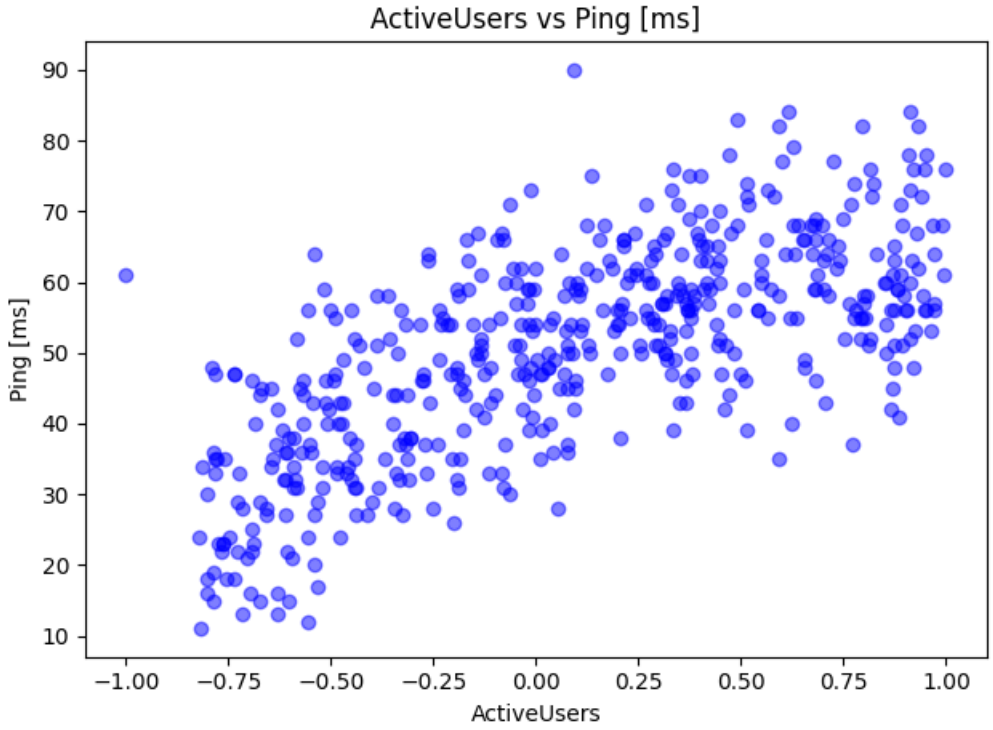
Začnu tím, že si data vizualizuji pro lepší představu toho, s čím budu vlastně pracovat.

Zřejmě budu pracovat se závislostí odezvy na počtu aktivní uživatelů na mé sociální síti.

Na grafu je možné si pouhým okem všimnout rostoucí tendence - tedy čím víc je uživatelů, tím větší je ping.

In [11]:
```
# ActiveUsers vs Ping [ms]
plt.scatter(df['ActiveUsers'], df['Ping'], alpha=0.5, color='blue')
plt.title('ActiveUsers vs Ping [ms]')
plt.xlabel('ActiveUsers')
plt.ylabel('Ping [ms]')


plt.tight_layout()
plt.show()
```



Teďka už můžu vytvořit formuli, která bude na začátku vypadat takto:

$$formule = \beta_0 + \beta_1 \cdot ActiveUsers + \beta_2 \cdot InteractingPct + \beta_3 \cdot ScrollingPct + \beta_4 \cdot MacOS + \beta_5 \cdot Windows + \beta_6 \cdot iOS + \beta_7 \cdot ActiveUsers^2 + \beta_8 \cdot Actiu$$
$$+ \beta_{12} ActiveUsers \cdot iOS + \beta_{13} \cdot InteractingPct^2 + \beta_{14} \cdot InteractingPct \cdot ScrollingPct + \beta_{15} \cdot InteractingPct \cdot MacOS + \beta_{16} \cdot InteractingPct \cdot Windc$$
$$\cdot ScrollingPct \cdot iOS$$

In [12]:
```
formula = (
    'Ping ~ ActiveUsers + InteractingPct + ScrollingPct + MacOS + Windows + iOS +'
    'I(ActiveUsers**2) + ActiveUsers:InteractingPct + ActiveUsers:ScrollingPct + \
        ActiveUsers:MacOS + ActiveUsers:Windows + ActiveUsers:iOS +'
    'I(InteractingPct**2) + InteractingPct:ScrollingPct + InteractingPct:MacOS + \
        InteractingPct:Windows + InteractingPct:iOS +'
    'I(ScrollingPct**2) + ScrollingPct:MacOS + ScrollingPct:Windows + ScrollingPct:iOS'
)

def evaluate_model(df, formula):
    model = smf.ols(formula=formula, data=df)
    result = model.fit()
    return result, model

result, model = evaluate_model(df, formula)
print(result.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:	16:38:14	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	30.0305	0.518	58.005	0.000	29.013	31.048
MacOS[T.True]	5.9228	0.522	11.337	0.000	4.896	6.949
Windows[T.True]	2.7871	0.524	5.317	0.000	1.757	3.817
iOS[T.True]	-3.5082	0.539	-6.514	0.000	-4.566	-2.450
ActiveUsers	13.9389	0.773	18.030	0.000	12.420	15.458
ActiveUsers:MacOS[T.True]	6.8470	1.509	4.536	0.000	3.881	9.813
ActiveUsers:Windows[T.True]	-3.7332	1.490	-2.505	0.013	-6.661	-0.806
ActiveUsers:iOS[T.True]	-5.1859	1.539	-3.369	0.001	-8.211	-2.161
InteractingPct	19.5866	0.568	34.488	0.000	18.471	20.702
InteractingPct:MacOS[T.True]	2.7831	1.293	2.153	0.032	0.243	5.323
InteractingPct:Windows[T.True]	1.6066	1.406	1.143	0.254	-1.155	4.368
InteractingPct:iOS[T.True]	-1.6202	1.400	-1.157	0.248	-4.372	1.131
ScrollingPct	10.4439	0.544	19.209	0.000	9.376	11.512
ScrollingPct:MacOS[T.True]	3.1397	1.290	2.433	0.015	0.604	5.675
ScrollingPct:Windows[T.True]	1.1805	1.365	0.865	0.387	-1.501	3.862
ScrollingPct:iOS[T.True]	-1.8880	1.344	-1.405	0.161	-4.528	0.752
I(ActiveUsers ** 2)	-10.0114	1.057	-9.469	0.000	-12.089	-7.934
ActiveUsers:InteractingPct	-0.5920	0.984	-0.601	0.548	-2.526	1.342
ActiveUsers:ScrollingPct	14.5309	0.949	15.308	0.000	12.666	16.396
I(InteractingPct ** 2)	8.3345	1.210	6.890	0.000	5.958	10.711
InteractingPct:ScrollingPct	11.2521	1.287	8.743	0.000	8.723	13.781
I(ScrollingPct ** 2)	-0.8082	1.214	-0.666	0.506	-3.194	1.577

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.	4.83e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.67e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [13]:

```
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 ====')
print(vif_df)

print('\n\n\n==== Matice korelace ====')
print(X.corr())
```

```
====  VIF ====
                                VIF
Intercept                  0.000000
MacOS[T.True]                   inf
Windows[T.True]                 inf
iOS[T.True]                     inf
ActiveUsers                     inf
ActiveUsers:MacOS[T.True]  2.385958
ActiveUsers:Windows[T.True]  2.434729
ActiveUsers:iOS[T.True]    2.231460
InteractingPct                  inf
InteractingPct:MacOS[T.True]    inf
InteractingPct:Windows[T.True]  inf
InteractingPct:iOS[T.True]      inf
ScrollingPct                    inf
ScrollingPct:MacOS[T.True]      inf
ScrollingPct:Windows[T.True]    inf
ScrollingPct:iOS[T.True]        inf
I(ActiveUsers ** 2)        1.145080
ActiveUsers:InteractingPct      inf
ActiveUsers:ScrollingPct        inf
I(InteractingPct ** 2)          inf
InteractingPct:ScrollingPct     inf
I(ScrollingPct ** 2)            inf



==== Matice korelace ====
                           Intercept  MacOS[T.True]  Windows[T.True]  \
Intercept                        NaN            NaN              NaN
MacOS[T.True]                    NaN       1.000000        -0.371550
Windows[T.True]                  NaN      -0.371550         1.000000
iOS[T.True]                      NaN      -0.341322        -0.334506
ActiveUsers                      NaN      -0.000136         0.003135
ActiveUsers:MacOS[T.True]        NaN       0.145366        -0.054011
ActiveUsers:Windows[T.True]      NaN      -0.053966         0.145245
ActiveUsers:iOS[T.True]          NaN      -0.016383        -0.016056
InteractingPct                   NaN       0.086466        -0.016964
InteractingPct:MacOS[T.True]     NaN       0.821075        -0.305071
InteractingPct:Windows[T.True]   NaN      -0.309767         0.833716
InteractingPct:iOS[T.True]       NaN      -0.272791        -0.267344
ScrollingPct                     NaN      -0.086466         0.016964
ScrollingPct:MacOS[T.True]       NaN       0.786822        -0.292344
ScrollingPct:Windows[T.True]     NaN      -0.316895         0.852901
ScrollingPct:iOS[T.True]         NaN      -0.289004        -0.283233
I(ActiveUsers ** 2)              NaN      -0.019938         0.027729
ActiveUsers:InteractingPct       NaN      -0.010314         0.021543
ActiveUsers:ScrollingPct         NaN       0.009561        -0.015185
I(InteractingPct ** 2)           NaN       0.108126        -0.042324
InteractingPct:ScrollingPct      NaN      -0.094711         0.101975
I(ScrollingPct ** 2)             NaN      -0.059509        -0.009201
```

	iOS[T.True]	ActiveUsers	\
Intercept	NaN	NaN	
MacOS[T.True]	-0.341322	-0.000136	
Windows[T.True]	-0.334506	0.003135	
iOS[T.True]	1.000000	-0.063206	
ActiveUsers	-0.063206	1.000000	
ActiveUsers:MacOS[T.True]	-0.049617	0.510339	
ActiveUsers:Windows[T.True]	-0.048585	0.522706	
ActiveUsers:iOS[T.True]	0.047999	0.490173	
InteractingPct	-0.062634	0.040275	
InteractingPct:MacOS[T.True]	-0.280251	-0.004409	
InteractingPct:Windows[T.True]	-0.278883	0.035105	
InteractingPct:iOS[T.True]	0.799219	-0.045884	
ScrollingPct	0.062634	-0.040275	
ScrollingPct:MacOS[T.True]	-0.268560	0.004539	
ScrollingPct:Windows[T.True]	-0.285300	-0.028045	
ScrollingPct:iOS[T.True]	0.846719	-0.057617	
I(ActiveUsers ** 2)	0.006416	0.331184	
ActiveUsers:InteractingPct	-0.063445	0.852873	
ActiveUsers:ScrollingPct	-0.045847	0.868594	
I(InteractingPct ** 2)	-0.053518	0.025793	
InteractingPct:ScrollingPct	-0.029387	0.052947	
I(ScrollingPct ** 2)	0.067640	-0.052018	

	ActiveUsers:MacOS[T.True]	\
Intercept	NaN	
MacOS[T.True]	0.145366	
Windows[T.True]	-0.054011	
iOS[T.True]	-0.049617	
ActiveUsers	0.510339	
ActiveUsers:MacOS[T.True]	1.000000	
ActiveUsers:Windows[T.True]	-0.007845	
ActiveUsers:iOS[T.True]	-0.002382	
InteractingPct	0.004535	
InteractingPct:MacOS[T.True]	0.111115	
InteractingPct:Windows[T.True]	-0.045030	
InteractingPct:iOS[T.True]	-0.039655	
ScrollingPct	-0.004535	
ScrollingPct:MacOS[T.True]	0.123288	
ScrollingPct:Windows[T.True]	-0.046066	
ScrollingPct:iOS[T.True]	-0.042011	
I(ActiveUsers ** 2)	0.124174	
ActiveUsers:InteractingPct	0.473606	
ActiveUsers:ScrollingPct	0.406879	
I(InteractingPct ** 2)	-0.001938	
InteractingPct:ScrollingPct	0.025090	
I(ScrollingPct ** 2)	-0.010645	

	ActiveUsers:Windows[T.True]	\
Intercept	NaN	
MacOS[T.True]	-0.053966	
Windows[T.True]	0.145245	

```
iOS[T.True]                                  -0.048585
ActiveUsers                                   0.522706
ActiveUsers:MacOS[T.True]                    -0.007845
ActiveUsers:Windows[T.True]                   1.000000
ActiveUsers:iOS[T.True]                      -0.002332
InteractingPct                                0.050021
InteractingPct:MacOS[T.True]                 -0.044310
InteractingPct:Windows[T.True]                0.181994
InteractingPct:iOS[T.True]                   -0.038830
ScrollingPct                                 -0.050021
ScrollingPct:MacOS[T.True]                   -0.042461
ScrollingPct:Windows[T.True]                  0.066301
ScrollingPct:iOS[T.True]                     -0.041138
I(ActiveUsers ** 2)                           0.212328
ActiveUsers:InteractingPct                    0.448325
ActiveUsers:ScrollingPct                      0.451625
I(InteractingPct ** 2)                        0.034562
InteractingPct:ScrollingPct                   0.055775
I(ScrollingPct ** 2)                         -0.062106

                             ActiveUsers:iOS[T.True]  InteractingPct  \
Intercept                                        NaN            NaN
MacOS[T.True]                              -0.016383       0.086466
Windows[T.True]                            -0.016056      -0.016964
iOS[T.True]                                 0.047999      -0.062634
ActiveUsers                                 0.490173       0.040275
ActiveUsers:MacOS[T.True]                  -0.002382       0.004535
ActiveUsers:Windows[T.True]                -0.002332       0.050021
ActiveUsers:iOS[T.True]                     1.000000       0.004645
InteractingPct                              0.004645       1.000000
InteractingPct:MacOS[T.True]               -0.013452       0.388633
InteractingPct:Windows[T.True]             -0.013386       0.248636
InteractingPct:iOS[T.True]                  0.047731       0.244987
ScrollingPct                               -0.004645      -1.000000
ScrollingPct:MacOS[T.True]                 -0.012891      -0.275403
ScrollingPct:Windows[T.True]               -0.013694      -0.262914
ScrollingPct:iOS[T.True]                    0.032349      -0.314208
I(ActiveUsers ** 2)                         0.108723      -0.014372
ActiveUsers:InteractingPct                  0.377028       0.100935
ActiveUsers:ScrollingPct                    0.464698      -0.028212
I(InteractingPct ** 2)                     -0.009932       0.967446
InteractingPct:ScrollingPct                 0.057095       0.022835
I(ScrollingPct ** 2)                       -0.018764      -0.968174

                             InteractingPct:MacOS[T.True]  ...  \
Intercept                                             NaN  ...
MacOS[T.True]                                    0.821075  ...
Windows[T.True]                                 -0.305071  ...
iOS[T.True]                                     -0.280251  ...
ActiveUsers                                     -0.004409  ...
ActiveUsers:MacOS[T.True]                        0.111115  ...
ActiveUsers:Windows[T.True]                     -0.044310  ...
```

```
ActiveUsers:iOS[T.True]                          -0.013452  ...
InteractingPct                                    0.388633  ...
InteractingPct:MacOS[T.True]                      1.000000  ...
InteractingPct:Windows[T.True]                   -0.254342  ...
InteractingPct:iOS[T.True]                       -0.223982  ...
ScrollingPct                                     -0.388633  ...
ScrollingPct:MacOS[T.True]                        0.293742  ...
ScrollingPct:Windows[T.True]                     -0.260195  ...
ScrollingPct:iOS[T.True]                         -0.237294  ...
I(ActiveUsers ** 2)                              -0.020475  ...
ActiveUsers:InteractingPct                        0.006596  ...
ActiveUsers:ScrollingPct                         -0.013658  ...
I(InteractingPct ** 2)                            0.399878  ...
InteractingPct:ScrollingPct                      -0.085526  ...
I(ScrollingPct ** 2)                             -0.352632  ...

                                 ScrollingPct  ScrollingPct:MacOS[T.True]  \
Intercept                                 NaN                         NaN
MacOS[T.True]                       -0.086466                    0.786822
Windows[T.True]                      0.016964                   -0.292344
iOS[T.True]                          0.062634                   -0.268560
ActiveUsers                         -0.040275                    0.004539
ActiveUsers:MacOS[T.True]           -0.004535                    0.123288
ActiveUsers:Windows[T.True]         -0.050021                   -0.042461
ActiveUsers:iOS[T.True]             -0.004645                   -0.012891
InteractingPct                      -1.000000                   -0.275403
InteractingPct:MacOS[T.True]        -0.388633                    0.293742
InteractingPct:Windows[T.True]      -0.248636                   -0.243732
InteractingPct:iOS[T.True]          -0.244987                   -0.214638
ScrollingPct                         1.000000                    0.275403
ScrollingPct:MacOS[T.True]           0.275403                    1.000000
ScrollingPct:Windows[T.True]         0.262914                   -0.249340
ScrollingPct:iOS[T.True]             0.314208                   -0.227395
I(ActiveUsers ** 2)                  0.014372                   -0.011249
ActiveUsers:InteractingPct          -0.100935                   -0.024403
ActiveUsers:ScrollingPct             0.028212                    0.030777
I(InteractingPct ** 2)              -0.967446                   -0.251289
InteractingPct:ScrollingPct         -0.022835                   -0.066129
I(ScrollingPct ** 2)                 0.968174                    0.281618

                                 ScrollingPct:Windows[T.True]  \
Intercept                                                 NaN
MacOS[T.True]                                       -0.316895
Windows[T.True]                                      0.852901
iOS[T.True]                                         -0.285300
ActiveUsers                                         -0.028045
ActiveUsers:MacOS[T.True]                           -0.046066
ActiveUsers:Windows[T.True]                          0.066301
ActiveUsers:iOS[T.True]                             -0.013694
InteractingPct                                      -0.262914
InteractingPct:MacOS[T.True]                        -0.260195
InteractingPct:Windows[T.True]                       0.422792
```

```
InteractingPct:iOS[T.True]                        -0.228018
ScrollingPct                                       0.262914
ScrollingPct:MacOS[T.True]                        -0.249340
ScrollingPct:Windows[T.True]                       1.000000
ScrollingPct:iOS[T.True]                          -0.241569
I(ActiveUsers ** 2)                                0.016313
ActiveUsers:InteractingPct                        -0.021580
ActiveUsers:ScrollingPct                          -0.026579
I(InteractingPct ** 2)                            -0.274007
InteractingPct:ScrollingPct                        0.071625
I(ScrollingPct ** 2)                               0.235113

                           ScrollingPct:iOS[T.True]  I(ActiveUsers ** 2)  \
Intercept                                       NaN                  NaN
MacOS[T.True]                             -0.289004            -0.019938
Windows[T.True]                           -0.283233             0.027729
iOS[T.True]                                0.846719             0.006416
ActiveUsers                               -0.057617             0.331184
ActiveUsers:MacOS[T.True]                 -0.042011             0.124174
ActiveUsers:Windows[T.True]               -0.041138             0.212328
ActiveUsers:iOS[T.True]                    0.032349             0.108723
InteractingPct                            -0.314208            -0.014372
InteractingPct:MacOS[T.True]              -0.237294            -0.020475
InteractingPct:Windows[T.True]            -0.236136             0.030878
InteractingPct:iOS[T.True]                 0.356937            -0.014302
ScrollingPct                               0.314208             0.014372
ScrollingPct:MacOS[T.True]                -0.227395            -0.011249
ScrollingPct:Windows[T.True]              -0.241569             0.016313
ScrollingPct:iOS[T.True]                   1.000000             0.022631
I(ActiveUsers ** 2)                        0.022631             1.000000
ActiveUsers:InteractingPct                -0.050894             0.301366
ActiveUsers:ScrollingPct                  -0.048381             0.269718
I(InteractingPct ** 2)                    -0.292046            -0.022238
InteractingPct:ScrollingPct               -0.054312             0.032594
I(ScrollingPct ** 2)                       0.316008             0.005672

                           ActiveUsers:InteractingPct  \
Intercept                                         NaN
MacOS[T.True]                               -0.010314
Windows[T.True]                              0.021543
iOS[T.True]                                 -0.063445
ActiveUsers                                  0.852873
ActiveUsers:MacOS[T.True]                    0.473606
ActiveUsers:Windows[T.True]                  0.448325
ActiveUsers:iOS[T.True]                      0.377028
InteractingPct                               0.100935
InteractingPct:MacOS[T.True]                 0.006596
InteractingPct:Windows[T.True]               0.060220
InteractingPct:iOS[T.True]                  -0.053899
ScrollingPct                                -0.100935
ScrollingPct:MacOS[T.True]                  -0.024403
ScrollingPct:Windows[T.True]                -0.021580
```

ScrollingPct:iOS[T.True]	-0.050894
I(ActiveUsers ** 2)	0.301366
ActiveUsers:InteractingPct	1.000000
ActiveUsers:ScrollingPct	0.482077
I(InteractingPct ** 2)	0.085955
InteractingPct:ScrollingPct	0.048500
I(ScrollingPct ** 2)	-0.109287

	ActiveUsers:ScrollingPct
Intercept	NaN
MacOS[T.True]	0.009561
Windows[T.True]	-0.015185
iOS[T.True]	-0.045847
ActiveUsers	0.868594
ActiveUsers:MacOS[T.True]	0.406879
ActiveUsers:Windows[T.True]	0.451625
ActiveUsers:iOS[T.True]	0.464698
InteractingPct	-0.028212
InteractingPct:MacOS[T.True]	-0.013658
InteractingPct:Windows[T.True]	0.001754
InteractingPct:iOS[T.True]	-0.025840
ScrollingPct	0.028212
ScrollingPct:MacOS[T.True]	0.030777
ScrollingPct:Windows[T.True]	-0.026579
ScrollingPct:iOS[T.True]	-0.048381
I(ActiveUsers ** 2)	0.269718
ActiveUsers:InteractingPct	0.482077
ActiveUsers:ScrollingPct	1.000000
I(InteractingPct ** 2)	-0.038295
InteractingPct:ScrollingPct	0.042816
I(ScrollingPct ** 2)	0.016434

	I(InteractingPct ** 2)
Intercept	NaN
MacOS[T.True]	0.108126
Windows[T.True]	-0.042324
iOS[T.True]	-0.053518
ActiveUsers	0.025793
ActiveUsers:MacOS[T.True]	-0.001938
ActiveUsers:Windows[T.True]	0.034562
ActiveUsers:iOS[T.True]	-0.009932
InteractingPct	0.967446
InteractingPct:MacOS[T.True]	0.399878
InteractingPct:Windows[T.True]	0.216347
InteractingPct:iOS[T.True]	0.235957
ScrollingPct	-0.967446
ScrollingPct:MacOS[T.True]	-0.251289
ScrollingPct:Windows[T.True]	-0.274007
ScrollingPct:iOS[T.True]	-0.292046
I(ActiveUsers ** 2)	-0.022238
ActiveUsers:InteractingPct	0.085955
ActiveUsers:ScrollingPct	-0.038295

```
I(InteractingPct ** 2)                       1.000000
InteractingPct:ScrollingPct                 -0.230920
I(ScrollingPct ** 2)                        -0.873316

                                 InteractingPct:ScrollingPct  \
Intercept                                            NaN
MacOS[T.True]                                  -0.094711
Windows[T.True]                                 0.101975
iOS[T.True]                                    -0.029387
ActiveUsers                                     0.052947
ActiveUsers:MacOS[T.True]                       0.025090
ActiveUsers:Windows[T.True]                     0.055775
ActiveUsers:iOS[T.True]                         0.057095
InteractingPct                                  0.022835
InteractingPct:MacOS[T.True]                   -0.085526
InteractingPct:Windows[T.True]                  0.101253
InteractingPct:iOS[T.True]                      0.009760
ScrollingPct                                   -0.022835
ScrollingPct:MacOS[T.True]                     -0.066129
ScrollingPct:Windows[T.True]                    0.071625
ScrollingPct:iOS[T.True]                       -0.054312
I(ActiveUsers ** 2)                             0.032594
ActiveUsers:InteractingPct                      0.048500
ActiveUsers:ScrollingPct                        0.042816
I(InteractingPct ** 2)                         -0.230920
InteractingPct:ScrollingPct                     1.000000
I(ScrollingPct ** 2)                           -0.272320

                                 I(ScrollingPct ** 2)
Intercept                                         NaN
MacOS[T.True]                               -0.059509
Windows[T.True]                             -0.009201
iOS[T.True]                                  0.067640
ActiveUsers                                 -0.052018
ActiveUsers:MacOS[T.True]                   -0.010645
ActiveUsers:Windows[T.True]                 -0.062106
ActiveUsers:iOS[T.True]                     -0.018764
InteractingPct                              -0.968174
InteractingPct:MacOS[T.True]                -0.352632
InteractingPct:Windows[T.True]              -0.264649
InteractingPct:iOS[T.True]                  -0.238233
ScrollingPct                                 0.968174
ScrollingPct:MacOS[T.True]                   0.281618
ScrollingPct:Windows[T.True]                 0.235113
ScrollingPct:iOS[T.True]                     0.316008
I(ActiveUsers ** 2)                          0.005672
ActiveUsers:InteractingPct                  -0.109287
ActiveUsers:ScrollingPct                     0.016434
I(InteractingPct ** 2)                      -0.873316
InteractingPct:ScrollingPct                 -0.272320
I(ScrollingPct ** 2)                         1.000000
```

```
[22 rows x 22 columns]
```

```
C:\Users\ondre\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\statsmodels\regression\linear_model.py:1782: RuntimeWarning: divide by zero encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
C:\Users\ondre\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\statsmodels\stats\outliers_influence.py:197: RuntimeWarning: divide by zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

Na výsledku si lze všimnout, že některé hodnoty VIFu jsou `> 10`.

Vytvořím si ještě matici korelace, abych zjistil, kde se nacházejí závislosti, které mi tvoří problémy s vytvořením modelu.

Zjišťuji, že problém tvoří korelace mezi `ScrollingPct` a `InteractingPct`. Dává to i smysl, vzhledem k tomu, že hodnoty si jsou doplňky, protože se jedná o procentuální hodnoty. Zbavit se tedy nejspíš mohu jedné z nich a je jedno které. Zkusím se zbavit `InteractingPct` a výpočet VIFu provedu znovu.

In [14]:
```
formula = (
    'Ping ~ ActiveUsers + InteractingPct + MacOS + Windows + iOS +'
    'I(ActiveUsers**2) + ActiveUsers:InteractingPct + ActiveUsers:MacOS + \
        ActiveUsers:Windows + ActiveUsers:iOS +'
    'I(InteractingPct**2) + InteractingPct:MacOS + InteractingPct:Windows + \
        InteractingPct:iOS'
)

result, model = evaluate_model(df, formula)
print(result.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:                        16:38:14   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                        39.6663      1.280     30.983      0.000      37.151      42.182
MacOS[T.True]                     9.0626      1.485      6.102      0.000       6.145      11.981
Windows[T.True]                   3.9677      1.516      2.617      0.009       0.989       6.947
iOS[T.True]                      -5.3962      1.494     -3.613      0.000      -8.331      -2.461
ActiveUsers                      28.4698      1.424     19.987      0.000      25.671      31.269
ActiveUsers:MacOS[T.True]         6.8470      1.509      4.536      0.000       3.881       9.813
ActiveUsers:Windows[T.True]      -3.7332      1.490     -2.505      0.013      -6.661      -0.806
ActiveUsers:iOS[T.True]          -5.1859      1.539     -3.369      0.001      -8.211      -2.161
InteractingPct                   22.0111      4.023      5.471      0.000      14.106      29.916
InteractingPct:MacOS[T.True]     -0.3566      2.530     -0.141      0.888      -5.327       4.614
InteractingPct:Windows[T.True]    0.4260      2.721      0.157      0.876      -4.919       5.771
InteractingPct:iOS[T.True]        0.2678      2.691      0.100      0.921      -5.020       5.556
I(ActiveUsers ** 2)             -10.0114      1.057     -9.469      0.000     -12.089      -7.934
ActiveUsers:InteractingPct      -15.1229      1.772     -8.532      0.000     -18.606     -11.640
I(InteractingPct ** 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.                         28.0
==============================================================================

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

Formuli jsem upravil tak, abych z ní vyškrtl `ScrollingPct`, teď zkusím znovu přepočítat VIFy.

In [15]:
```python
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 ====')
print(vif_df)
```

```
====  VIF ====
                                    VIF
Intercept                         23.383991
MacOS[T.True]                      6.272004
Windows[T.True]                    6.417808
iOS[T.True]                        5.723560
ActiveUsers                        7.817225
ActiveUsers:MacOS[T.True]          2.385958
ActiveUsers:Windows[T.True]        2.434729
ActiveUsers:iOS[T.True]            2.231460
InteractingPct                    20.191723
InteractingPct:MacOS[T.True]       7.586537
InteractingPct:Windows[T.True]     6.857633
InteractingPct:iOS[T.True]         6.028016
I(ActiveUsers ** 2)                1.145080
ActiveUsers:InteractingPct         3.872128
I(InteractingPct ** 2)            16.060875
```

Nyní už vycházejí všechny hodnoty VIFu `< 10`, což je přijatelné. Můžu se tedy přesunout k zpětné eliminaci.

To znamená, že teď budu potřebovat, aby všechny hodnoty ve sloupci `P > |t|` byly menší, než jaký je můj nastavený práh. Ten bude nastaven na klasických `5%`, tedy `0.05`.

Toho docílím tak, že postupně budu odstraňovat největší členy formule a sledovat, jak se model změní při postupných změnách. V tuto chvíli je největší hodnota `InteractingPct:iOS[T.True]` s hodnotou `0.921`. Smažu tedy tento člen.

In [16]:
```
formula = (
    'Ping ~ ActiveUsers + InteractingPct + MacOS + Windows + iOS +'
    'I(ActiveUsers**2) + ActiveUsers:InteractingPct + ActiveUsers:MacOS + \
        ActiveUsers:Windows + ActiveUsers:iOS +'
    'I(InteractingPct**2) + InteractingPct:MacOS + InteractingPct:Windows'
)

result, model = evaluate_model(df, formula)
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     202.8
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          3.58e-187
Time:                        16:38:14   Log-Likelihood:                -1598.4
No. Observations:                 502   AIC:                             3225.
Df Residuals:                     488   BIC:                             3284.
Df Model:                          13
Covariance Type:            nonrobust
=====================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
Intercept                          39.5955      1.063     37.249      0.000      37.507      41.684
MacOS[T.True]                       9.1305      1.317      6.931      0.000       6.542      11.719
Windows[T.True]                     4.0360      1.350      2.989      0.003       1.383       6.689
iOS[T.True]                        -5.2709      0.802     -6.569      0.000      -6.847      -3.694
ActiveUsers                        28.4784      1.420     20.051      0.000      25.688      31.269
ActiveUsers:MacOS[T.True]           6.8517      1.507      4.546      0.000       3.891       9.813
ActiveUsers:Windows[T.True]        -3.7296      1.488     -2.506      0.013      -6.653      -0.806
ActiveUsers:iOS[T.True]            -5.1826      1.538     -3.371      0.001      -8.204      -2.162
InteractingPct                     22.1702      3.688      6.011      0.000      14.924      29.417
InteractingPct:MacOS[T.True]       -0.4987      2.086     -0.239      0.811      -4.597       3.600
InteractingPct:Windows[T.True]      0.2835      2.311      0.123      0.902      -4.257       4.824
I(ActiveUsers ** 2)               -10.0128      1.056     -9.481      0.000     -12.088      -7.938
ActiveUsers:InteractingPct        -15.1482      1.752     -8.644      0.000     -18.591     -11.705
I(InteractingPct ** 2)             -3.7412      3.485     -1.074      0.284     -10.589       3.106
==============================================================================
Omnibus:                      228.545   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3156.249
Skew:                           1.604   Prob(JB):                         0.00
Kurtosis:                      14.858   Cond. No.                         25.3
==============================================================================

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

První člen byl odstraněn, nicméně je potřeba pracovat dále. Nyní je největším členem `InteractingPct:Windows[T.True]` s hodnotou `0.902`. Tak jej odstraníme a opět vyhodnotíme.

In [17]:
```python
formula = (
    'Ping ~ ActiveUsers + InteractingPct + MacOS + Windows + iOS +'
    'I(ActiveUsers**2) + ActiveUsers:InteractingPct + ActiveUsers:MacOS + \
        ActiveUsers:Windows + ActiveUsers:iOS +'
    'I(InteractingPct**2) + InteractingPct:MacOS'
)

result, model = evaluate_model(df, formula)
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     220.1
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          2.38e-188
Time:                        16:38:14   Log-Likelihood:                -1598.4
No. Observations:                 502   AIC:                             3223.
Df Residuals:                     489   BIC:                             3278.
Df Model:                          12
Covariance Type:            nonrobust
=================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
Intercept                        39.5502      0.996     39.711      0.000      37.593      41.507
MacOS[T.True]                     9.1752      1.265      7.255      0.000       6.690      11.660
Windows[T.True]                   4.1710      0.782      5.332      0.000       2.634       5.708
iOS[T.True]                      -5.2687      0.801     -6.574      0.000      -6.843      -3.694
ActiveUsers                      28.4773      1.419     20.071      0.000      25.690      31.265
ActiveUsers:MacOS[T.True]         6.8548      1.505      4.554      0.000       3.897       9.812
ActiveUsers:Windows[T.True]      -3.7160      1.482     -2.507      0.013      -6.629      -0.803
ActiveUsers:iOS[T.True]          -5.1807      1.536     -3.373      0.001      -8.198      -2.163
InteractingPct                   22.2643      3.604      6.178      0.000      15.184      29.345
InteractingPct:MacOS[T.True]     -0.5923      1.940     -0.305      0.760      -4.403       3.219
I(ActiveUsers ** 2)             -10.0112      1.055     -9.490      0.000     -12.084      -7.938
ActiveUsers:InteractingPct      -15.1522      1.750     -8.657      0.000     -18.591     -11.713
I(InteractingPct ** 2)           -3.7414      3.482     -1.075      0.283     -10.582       3.099
==============================================================================
Omnibus:                      228.538   Durbin-Watson:                   1.932
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3155.564
Skew:                           1.604   Prob(JB):                         0.00
Kurtosis:                      14.856   Cond. No.                         24.8
==============================================================================

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

Stále jsou zde členy, které je potřeba zmenšit. Nyní to bude `InteractingPct:MacOS[T.True]` s hodnotou `0.760`.

In [18]:
```
formula = (
    'Ping ~ ActiveUsers + InteractingPct + MacOS + Windows + iOS +'
    'I(ActiveUsers**2) + ActiveUsers:InteractingPct + ActiveUsers:MacOS + \
        ActiveUsers:Windows + ActiveUsers:iOS +'
    'I(InteractingPct**2)'
)

result, model = evaluate_model(df, formula)
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     240.6
Date:                Sun, 15 Dec 2024   Prob (F-statistic):          1.57e-189
Time:                        16:38:14   Log-Likelihood:                -1598.5
No. Observations:                 502   AIC:                             3221.
Df Residuals:                     490   BIC:                             3272.
Df Model:                          11
Covariance Type:            nonrobust
=================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
Intercept                        39.6342      0.956     41.444      0.000      37.755      41.513
MacOS[T.True]                     8.8714      0.780     11.372      0.000       7.339      10.404
Windows[T.True]                   4.1703      0.781      5.337      0.000       2.635       5.706
iOS[T.True]                      -5.2725      0.801     -6.586      0.000      -6.845      -3.699
ActiveUsers                      28.4814      1.417     20.093      0.000      25.696      31.266
ActiveUsers:MacOS[T.True]         6.8512      1.504      4.556      0.000       3.896       9.806
ActiveUsers:Windows[T.True]      -3.7118      1.481     -2.506      0.013      -6.622      -0.802
ActiveUsers:iOS[T.True]          -5.1860      1.534     -3.380      0.001      -8.201      -2.171
InteractingPct                   22.1153      3.567      6.199      0.000      15.106      29.124
I(ActiveUsers ** 2)             -10.0161      1.054     -9.504      0.000     -12.087      -7.945
ActiveUsers:InteractingPct      -15.1459      1.749     -8.662      0.000     -18.581     -11.710
I(InteractingPct ** 2)           -3.7789      3.476     -1.087      0.278     -10.609       3.051
==============================================================================
Omnibus:                      229.759   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3209.574
Skew:                           1.611   Prob(JB):                         0.00
Kurtosis:                      14.961   Cond. No.                         24.6
==============================================================================

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

Opět vyhodnoceno, ale stále zbývá člen `I(InteractingPct ** 2)` s hodnotou `0.278`, který je potřeba odstranit.

In [19]:

```
formula = (
    'Ping ~ ActiveUsers + InteractingPct + MacOS + Windows + iOS +'
    'I(ActiveUsers**2) + ActiveUsers:InteractingPct + ActiveUsers:MacOS + \
        ActiveUsers:Windows + ActiveUsers:iOS'
)

result, model = evaluate_model(df, formula)
print(result.summary())

# REZIDUA VS PREDIKOVANE HODNOTY
plt.scatter(result.fittedvalues, result.resid, alpha=0.5, color='blue')

plt.axhline(y=0, color='r', linestyle='-')
plt.grid(True)
plt.xlabel('Predikované hodnoty')
plt.ylabel('Rezidua')
plt.title('Rezidua vs Predikované hodnoty')
plt.show()

# HISTOGRAM REZIDUI
plt.hist(result.resid, bins='auto', color='blue', alpha=0.5, density=True)

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, np.mean(result.resid), np.std(result.resid))
plt.plot(x, p, 'k', linewidth=2)
plt.grid(True)
plt.title("Histogram reziduí")
plt.xlabel("Rezidua")
plt.ylabel("Hustota")
plt.show()

# Q-Q GRAF
qqplot(result.resid, line='s')
plt.title('Q-Q graf')
plt.grid(True)
plt.show()
```

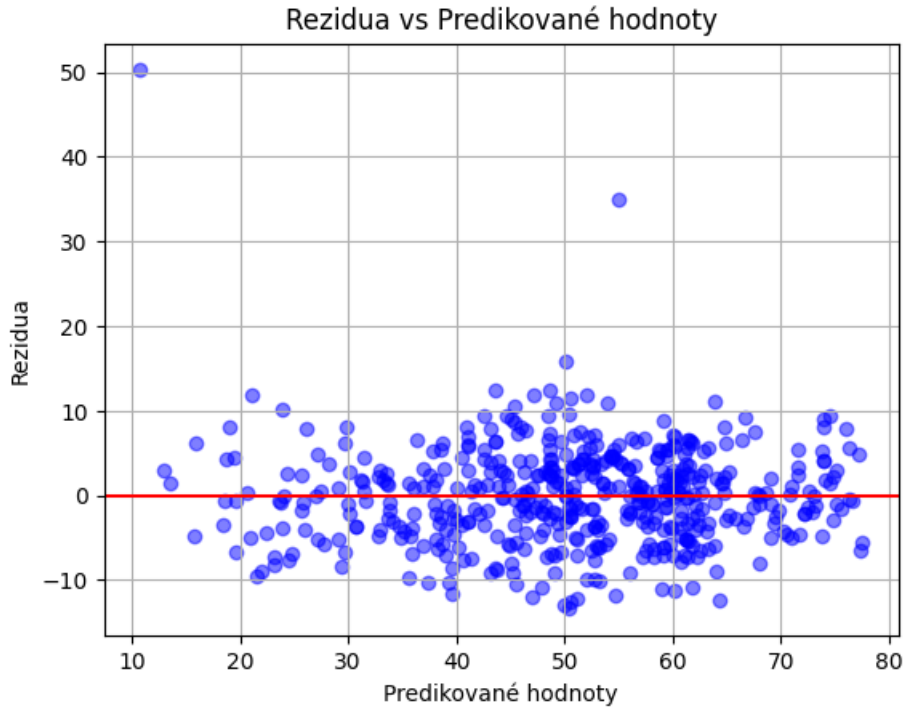
OLS Regression Results

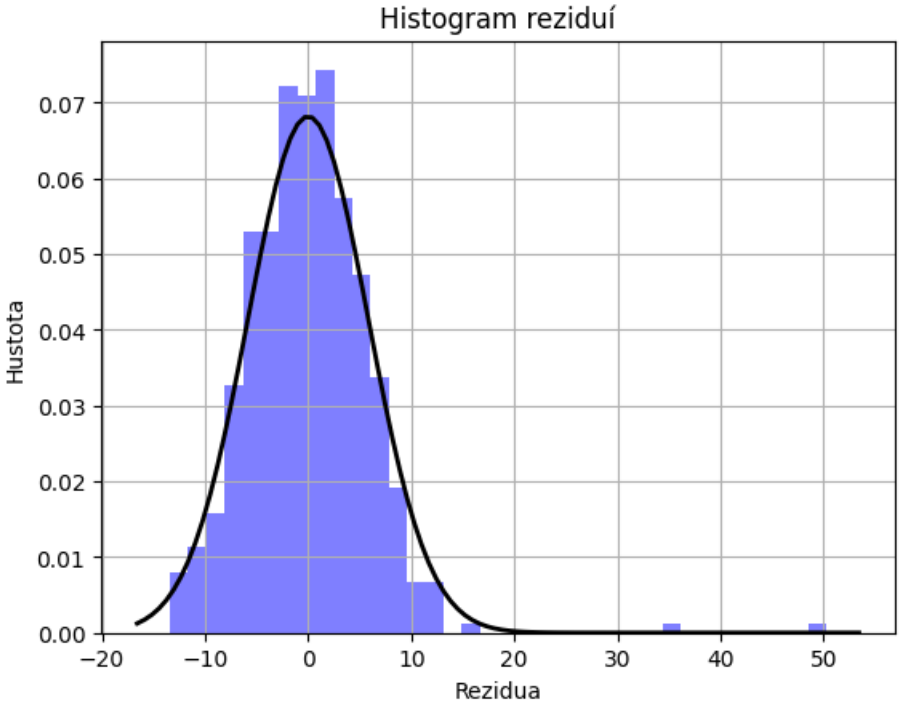
Dep. Variable:	Ping	R-squared:	0.843
Model:	OLS	Adj. R-squared:	0.840
Method:	Least Squares	F-statistic:	264.4
Date:	Sun, 15 Dec 2024	Prob (F-statistic):	1.69e-190
Time:	16:38:14	Log-Likelihood:	-1599.1
No. Observations:	502	AIC:	3220.
Df Residuals:	491	BIC:	3267.
Df Model:	10		
Covariance Type:	nonrobust		

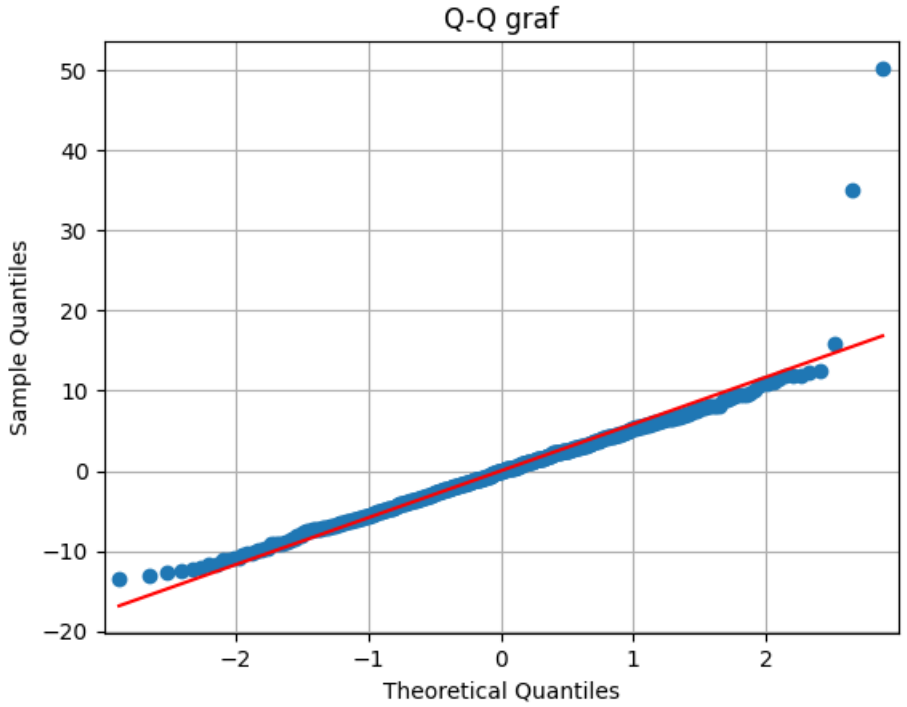
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	40.2495	0.771	52.206	0.000	38.735	41.764
MacOS[T.True]	8.7999	0.777	11.318	0.000	7.272	10.328
Windows[T.True]	4.1936	0.781	5.367	0.000	2.658	5.729
iOS[T.True]	-5.3112	0.800	-6.640	0.000	-6.883	-3.740
ActiveUsers	28.4186	1.417	20.062	0.000	25.635	31.202
ActiveUsers:MacOS[T.True]	6.9549	1.501	4.633	0.000	4.005	9.904
ActiveUsers:Windows[T.True]	-3.6108	1.478	-2.442	0.015	-6.515	-0.706
ActiveUsers:iOS[T.True]	-5.0588	1.530	-3.306	0.001	-8.065	-2.052
InteractingPct	18.3647	0.907	20.253	0.000	16.583	20.146
I(ActiveUsers ** 2)	-9.9907	1.054	-9.480	0.000	-12.061	-7.920
ActiveUsers:InteractingPct	-15.1484	1.749	-8.662	0.000	-18.585	-11.712

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.	11.9

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


Rezidua vs Predikované hodnoty
Rezidua
Predikované hodnoty
50
40
30
20
10
0
−10
10
20
30
40
50
60
70
80


Histogram reziduí
Hustota
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
−20
−10
0
10
20
30
40
50
Rezidua



Poslední úprava mě dostala ke zdárnému výsledku. Žádný z členů formule mi nepřesahuje hodnotu `0.05`, přičemž největší z nich je `ActiveUsers:Windows[T.True]` s hodnotou `0.015`, což je v pořádku.

Finální formule tedy vypadá takto:

$$formula = \beta_0 + \beta_1 \cdot ActiveUsers + \beta_2 \cdot InteractingPct + \beta_3 \cdot MacOS + \beta_4 \cdot Windows + \beta_5 \cdot iOS + \beta_6 \cdot ActiveUsers^2 + \beta_7 \cdot ActiveUsers \cdot Interactir$$

Rovnice modelu

Rovnice modelu tedy bude vypadat následovně:

$$model = 40.2495 + 24.4186 \cdot ActiveUsers + 18.3647 \cdot InteractingPct + 8.7999 \cdot MacOS + 4.1936 \cdot Windows - 5.3112 \cdot iOS - 9.9907 \cdot ActiveUsers^2 -$$
$$\cdot ActiveUsers \cdot Windows$$

Diskuze splnění předpokladů lineární regrese

Model má velice poměrně dobrou hodnotu **R-squared** (`0.843`), díky čemuž by měl být schopný solidně predikovat ping.

Výsledek **Durbin-Wattsonova** testu vyšel `1.928`, což znamená, že rezidua nejsou autokorelovaná, takže jejich závislost není problémem.

Problém je ovšem výsledek **Jarque-Berra** testu (`0.0`), podle kterého rezidua nejsou normálně rozložena. To v poměrně viditelně ukazuje i histogram reziduí výše. Je možné, že dělají problém odlehlé hodnoty, které teď mohou dost ovlivňovat přesnost modelu (spoiler: je to pravda).

Celkově by tedy model nebyl vůbec špatný, nicméně se v dalším bodu níže zaměřím na problém s outliery.

Zbavení se odlehlých hodnot

In [20]:
```
influence = result.get_influence()
# Leverage
leverage = influence.hat_matrix_diag
# Cookovy D hodnoty (a p-hodnoty) jako n-tice polí [n x 2]
cooks_d = influence.cooks_distance
# Standardizovaná rezidua
standardized_residuals = influence.resid_studentized_internal
# Studentizovaná rezidua
studentized_residuals = influence.resid_studentized_external
# Výpočet p-hodnot pro studentizovaná rezidua
studentized_residuals_pvalues = 2 * (1 - stats.t.cdf(np.abs(studentized_residuals), df=df.shape[0]-len(result.params)))

#tabulka dohromady dořešit změny počtu řádků
outl_stats_df = pd.DataFrame({
    'Leverage': leverage,
    'Standardized Residuals': standardized_residuals,
    'Studentized Residuals': studentized_residuals,
    'Studentized Residuals p-value': studentized_residuals_pvalues,
    'Cook\'s Distance': cooks_d[0],
    'Cook\'s Distance_p-value': cooks_d[1]
}, index=df.index)
#vyber jen "zajímavý" hodnoty
outl_stats_df = outl_stats_df[(outl_stats_df['Leverage'] > 3*len(result.params)/df.shape[0]) | \
                              (np.abs(outl_stats_df['Standardized Residuals']) > 2) | \
                                (outl_stats_df['Cook\'s Distance_p-value'] < 0.05)]

summary_frame = influence.summary_frame()

print(outl_stats_df)
```

```
     Leverage  Standardized Residuals  Studentized Residuals  \
62   0.012590               -2.036977              -2.043554
82   0.010646                2.699228               2.716710
114  0.012955                2.111260               2.118748
129  0.014222               -2.141213              -2.149089
145  0.023780               -2.292470              -2.302490
178  0.047086                2.054883               2.061673
254  0.011482                2.011917               2.018204
255  0.009986                5.945469               6.165493
310  0.016649               -2.111115              -2.118601
332  0.030075                2.124928               2.132592
428  0.028086                2.048785               2.055502
430  0.017414               -2.080739              -2.087844
476  0.074941                8.830417               9.618155
490  0.026903               -2.230330              -2.239431

     Studentized Residuals p-value  Cook's Distance  Cook's Distance_p-value
62                     4.153079e-02         0.004810                 1.000000
82                     6.826184e-03         0.007127                 1.000000
114                    3.461327e-02         0.005318                 1.000000
129                    3.211548e-02         0.006013                 1.000000
145                    2.172492e-02         0.011638                 1.000000
178                    3.976461e-02         0.018968                 1.000000
254                    4.411335e-02         0.004274                 1.000000
255                    1.467855e-09         0.032412                 1.000000
310                    3.462578e-02         0.006860                 1.000000
332                    3.345366e-02         0.012728                 1.000000
428                    4.035881e-02         0.011027                 1.000000
430                    3.732667e-02         0.006975                 1.000000
476                    0.000000e+00         0.574273                 0.850171
490                    2.557484e-02         0.012502                 1.000000
```

Podle leverage na první pohled nic nepoznám, nicméně jakmile se podívám na hodnoty reziduí, tak je vidět, že dvě hodnoty vybočují z nějakého standardu. To i poměrně potvrzují grafy reziduí v předchozím bodě (rezidua vs predikované hodnoty, histogram reziduí a Q-Q graf) Konkrétně to jsou hodnoty na indexu `255` a `476` (ve výpisu statistik výše).

Jelikož jsem si z grafů poměrně jistý, že to jsou právě tyto hodnoty, které dělají problém, přejdu k jejich smazání.

Nicméně ještě předtím bych ukázal, o která data se jedná na grafu, který jsem si vykresloval ještě "před dávnými časy v předaleké galaxii" (asi o 10-15 buněk výš).

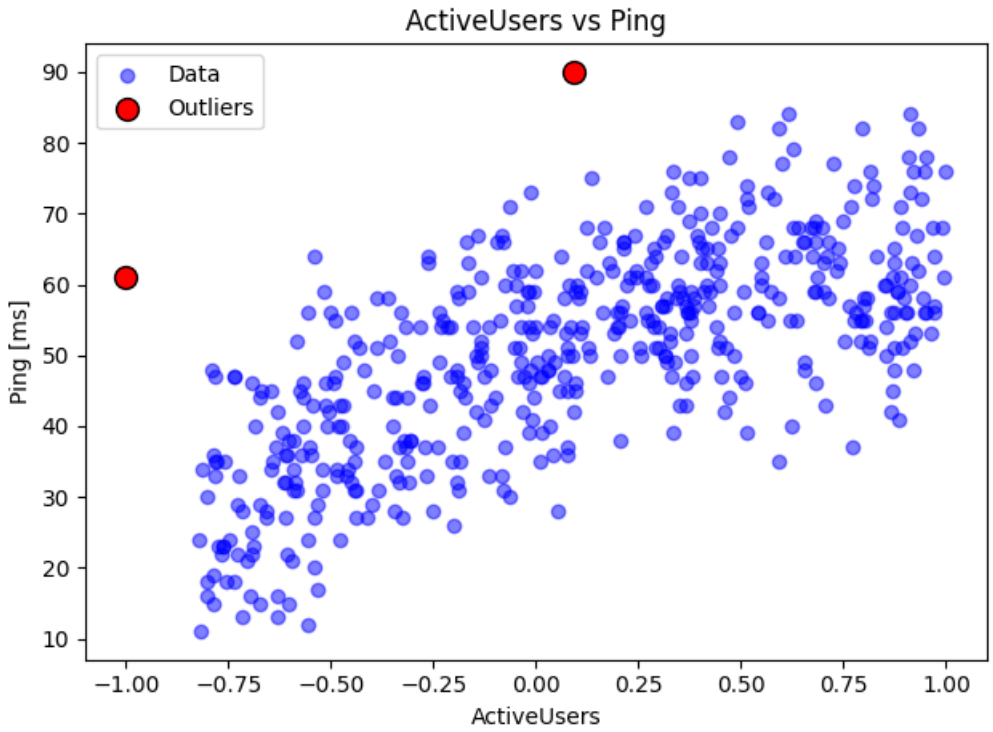
In [21]:
```
outliers = [255, 476]

# plt.figure(figsize=(10, 6))
plt.scatter(df['ActiveUsers'], df['Ping'], alpha=0.5, color='blue', label='Data')

plt.scatter(
    df.loc[outliers, 'ActiveUsers'],
    df.loc[outliers, 'Ping'],
    color='red',
    label='Outliers',
    s=100,
    edgecolors='black'
)

plt.title('ActiveUsers vs Ping')
plt.xlabel('ActiveUsers')
plt.ylabel('Ping [ms]')
plt.legend()

plt.tight_layout()
plt.show()
```



No, teď tedy přejdu k smazání těchto outlierů. Zase si také vykreslím grafy, jako předtím, čistě pro kontrolu, že jsem nikde neudělal chybu.

A zjevně nic špatně není. Graf reziduí je nyní již opravdu okolo nuly a obecný graf dat už neobsahuje zvýrazněné outliery, jako výše. Vše tedy zjevně funguje.

In [22]:
```
# vypustim outliery
df_final = df.drop(index=outliers)

# prepoctu model bez outlieru
result_final, model_final = evaluate_model(df_final, formula)

print(result_final.summary())

# GRAF REZIDUI VS PREDIKOVANE HODNOTY
plt.scatter(result_final.fittedvalues, result_final.resid, alpha=0.5, color='blue')

plt.axhline(y=0, color='r', linestyle='-')
plt.grid(True)
plt.xlabel('Predikované hodnoty')
plt.ylabel('Rezidua')
plt.title('Rezidua vs Predikované hodnoty')
plt.show()

# HISTOGRAM REZIDUI
plt.hist(result_final.resid, bins='auto', color='blue', alpha=0.5, density=True)

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, np.mean(result_final.resid), np.std(result_final.resid))
plt.plot(x, p, 'k', linewidth=2)
plt.grid(True)
plt.title("Histogram reziduí")
plt.xlabel("Rezidua")
plt.ylabel("Hustota")
plt.show()

# Q-Q GRAF
qqplot(result_final.resid, line='s')
plt.title('Q-Q graf')
plt.grid(True)
plt.show()

# GRAFIK DAT BEZ OUTLIERU
plt.scatter(df_final['ActiveUsers'], df_final['Ping'], alpha=0.5, color='blue', label='Data')

plt.title('ActiveUsers vs Ping')
plt.xlabel('ActiveUsers')
plt.ylabel('Ping [ms]')
plt.legend()

plt.tight_layout()
plt.show()
```

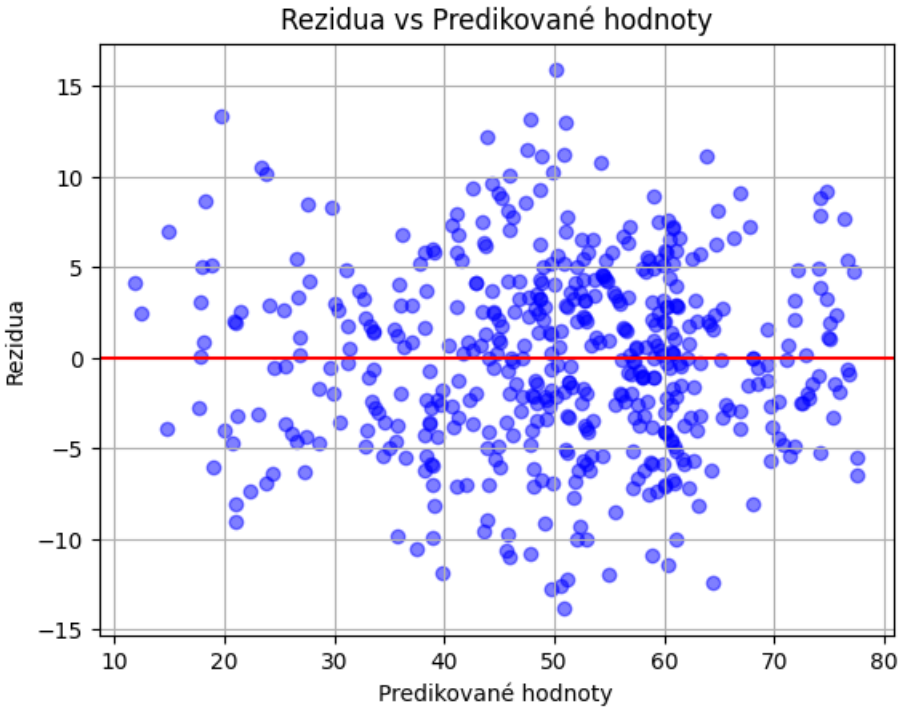
OLS Regression Results

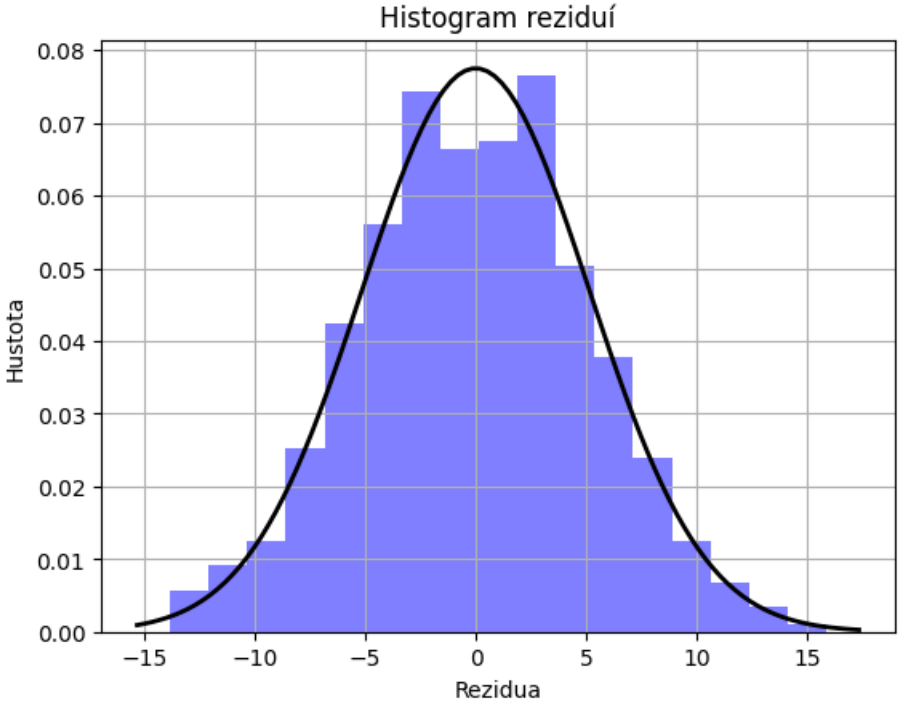
Dep. Variable:	Ping	R-squared:	0.877
Model:	OLS	Adj. R-squared:	0.875
Method:	Least Squares	F-statistic:	349.9
Date:	Sun, 15 Dec 2024	Prob (F-statistic):	1.28e-215
Time:	16:38:15	Log-Likelihood:	-1528.7
No. Observations:	500	AIC:	3079.
Df Residuals:	489	BIC:	3126.
Df Model:	10		
Covariance Type:	nonrobust		

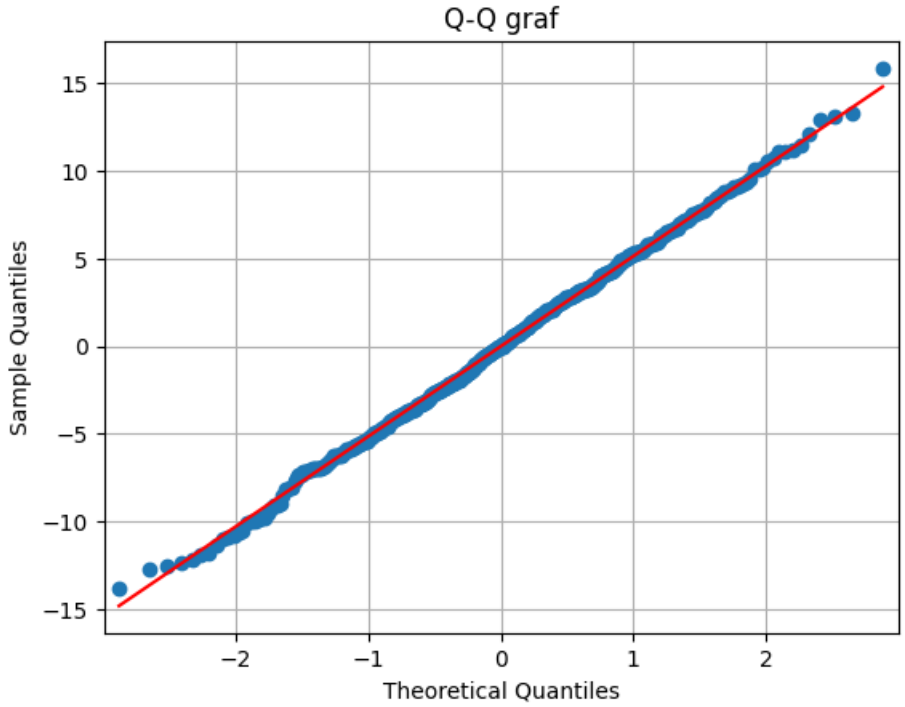
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	40.2973	0.679	59.389	0.000	38.964	41.630
MacOS[T.True]	8.2538	0.686	12.028	0.000	6.906	9.602
Windows[T.True]	3.9905	0.689	5.794	0.000	2.637	5.344
iOS[T.True]	-5.2566	0.704	-7.468	0.000	-6.640	-3.874
ActiveUsers	29.3976	1.251	23.499	0.000	26.940	31.856
ActiveUsers:MacOS[T.True]	8.5302	1.330	6.415	0.000	5.918	11.143
ActiveUsers:Windows[T.True]	-3.6770	1.301	-2.827	0.005	-6.233	-1.121
ActiveUsers:iOS[T.True]	-5.2718	1.347	-3.915	0.000	-7.918	-2.626
InteractingPct	18.8365	0.799	23.568	0.000	17.266	20.407
I(ActiveUsers ** 2)	-11.1165	0.940	-11.832	0.000	-12.962	-9.271
ActiveUsers:InteractingPct	-16.6230	1.546	-10.752	0.000	-19.661	-13.585

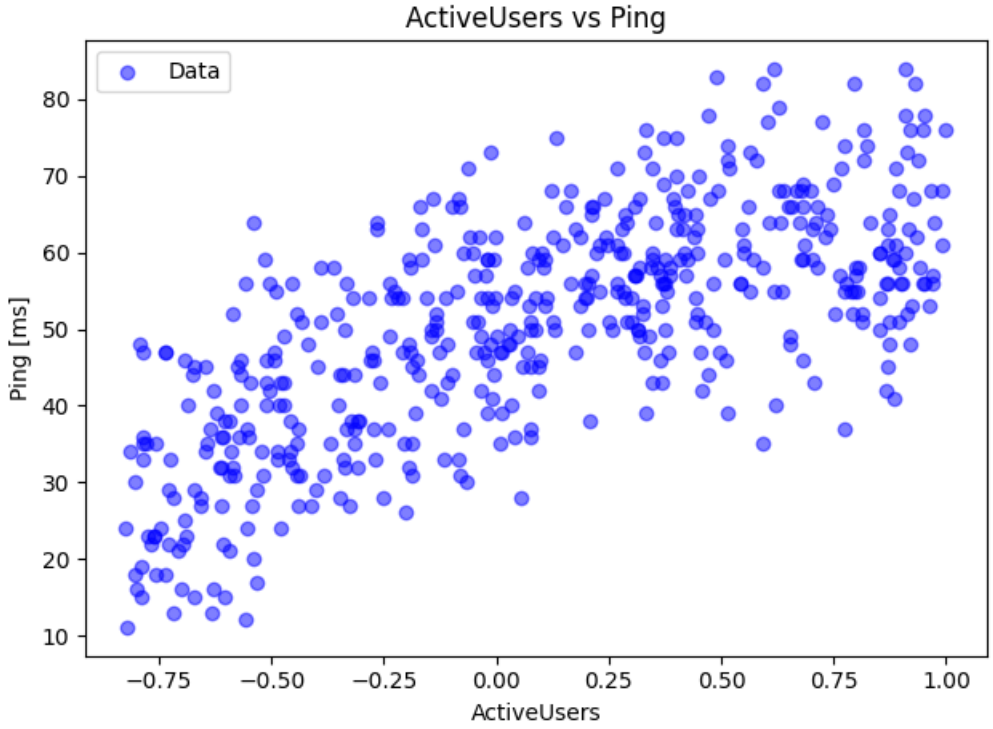
Omnibus:	0.661	Durbin-Watson:	1.990
Prob(Omnibus):	0.719	Jarque-Bera (JB):	0.750
Skew:	0.014	Prob(JB):	0.687
Kurtosis:	2.812	Cond. No.	11.9

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


Rezidua vs Predikované hodnoty
Rezidua
Predikované hodnoty
15
10
5
0
−5
−10
−15
10
20
30
40
50
60
70
80


Histogram reziduí
Hustota
Rezidua
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−15
−10
−5
0
5
10
15


Q-Q graf
Sample Quantiles
Theoretical Quantiles
15
10
5
0
−5
−10
−15
−2
−1
0
1
2



Jako shrnutí je dobré zmínit, že jak jsem zmiňoval ve diskuzi v předchozím bodu, graf opravdu obsahoval outliery. Těch jsem se úspěšně zbavil a nyní vypadá Q-Q graf a histogram reziduí výrazně lépe. I Jarque-Berra test, který v předchozím bodu naznačoval, že by odlehlé hodnoty mohly být problém, již ukazuje, že rezidua víceméně odpovídají normálnímu rozdělení.

2.2

Pomocí Vašeho výsledného modelu identifikujte, pro které nastavení parametrů má odezva nejproblematičtější (největší) hodnotu (použijte model, nikoli samotná pozorování).

In [23]:

```python
# opisu si rovnici modelu z vysledku prvniho bodu
def model_equation(params):
    active_users, interacting_pct, macos, windows, ios = params

    return - (
        + 40.2495
        + 24.4186 * active_users
        + 18.3647 * interacting_pct
        + 8.7999 * macos
        + 4.1936 * windows
        - 5.3112 * ios
        - 9.9907 * active_users**2
        - 15.1484 * active_users * interacting_pct
        + 6.9549 * active_users * macos
        - 3.6108 * active_users * windows
        - 5.0588 * active_users * ios
    )

# nastavim si pocatecni hodnoty na stred intervalu (neni to teda prumer hodnot, ale tak aspon cca)
# pouzil jsem totiz preskalovani na interval <-1; 1>
intial_guess = [0, 0, 0, 0, 1]

# nastavim si hranice, ve kterych se parametry muzou pohybovat (podle zvoleneho skalovani)
# MacOS, Windows, iOS jsou dummy promenne, takze ty jsou bud 0 nebo 1 (delal jsem to na zacatku tohoto ukolu)
bounds = [
    (-1, 1), # ActiveUsers
    (-1, 1), # InteractingPct
    (0, 1),  # MacOS
    (0, 1),  # Windows
    (0, 1)   # iOS
]

# nastavim si podminku, ze soucet MacOS, Windows a iOS musi byt 1
# muze tam byt max jeden z nich
# pokud jsou 0 vsichni, pak je to Android
constraints = [
    {'type': 'ineq', 'fun': lambda x: 1 - (x[2] + x[3] + x[4])}
]

# spustim optimalizaci (podobne jako v prvnim ukolu)
result2 = opt.minimize(model_equation, intial_guess, bounds=bounds, constraints=constraints)

# tady tohle mi da, jake jsou optimalni parametry
optimal_params = result2.x

# z tohodle zjistim, jaky je maximalni ping
max_ping = -result2.fun

# a nejak si to vypisu
print(f'Optimal parameters:')
print(f'\tActiveUsers: {round(optimal_params[0], 4)} => denormalized: ~ {round(denormalize_value(optimal_params[0]))}')
print(f'\tInteractingPct: {round(optimal_params[1], 4)}')
```

```
print(f'\tMacOS: {round(optimal_params[2])}')
print(f'\tWindows: {round(optimal_params[3])}')
print(f'\tiOS: {round(optimal_params[4])}')

print(f'\nMax ping: {round(max_ping, 4)}')
```

```
Optimal parameters:
        ActiveUsers: 0.812 => denormalized: ~ 9032
        InteractingPct: 1.0
        MacOS: 1
        Windows: 0
        iOS: 0

Max ping: 74.0016
```

2.3

Odhadněte hodnotu odezvy uživatele s Windows, při průměrném nastavení ostatních parametrů a vypočtěte konfidenční interval a predikční interval pro toto nastavení.

In [24]:
```python
# z normalizovaneho dataframe si vytahnu mean pro ActiveUsers a InteractingPct
mean_active = df_final['ActiveUsers'].mean()
mean_interacting = df_final['InteractingPct'].mean()

# vytvorim si dataframe s prumernymi hodnotami
tmp_df_mean = pd.DataFrame({
    'ActiveUsers': [mean_active],
    'InteractingPct': [mean_interacting],
    'MacOS': [0],
    'Windows': [1],
    'iOS': [0],
})

# z predikce si vytahnu hodnoty
predicted_windows = result_final.get_prediction(tmp_df_mean)
# musim vlozit konfidencni interval
predicted_windows_summary = predicted_windows.summary_frame(0.05)

predicted_ping = predicted_windows_summary.iloc[0]['mean']

# vytahnu si konfidencni interval
ci_lower = predicted_windows_summary.iloc[0]['mean_ci_lower']
ci_upper = predicted_windows_summary.iloc[0]['mean_ci_upper']

# a vytahnu i predikcni interval
pi_lower = predicted_windows_summary.iloc[0]['obs_ci_lower']
pi_upper = predicted_windows_summary.iloc[0]['obs_ci_upper']

# a vypisu vysledky...
# predikovany ping
print(f'Predicted ping: {round(predicted_ping, 4)}')

# konfidencni interval
print(f'Confidence interval (95%): <{round(ci_lower, 4)}; {round(ci_upper, 4)}>')

# predikcni interval
print(f'Prediction interval (95%): <{round(pi_lower, 4)}; {round(pi_upper, 4)}>')
```

```
Predicted ping: 55.0027
Confidence interval (95%): <53.9721; 56.0332>
Prediction interval (95%): <44.7247; 65.2807>
```

2.4

Na základě jakýchkoli vypočtených charakteristik argumentujte, zdali je váš model "vhodný" pro další použití.

In [25]:
```python
# vypisu si statistiky vysledku
print(result_final.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:	16:38:15	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	40.2973	0.679	59.389	0.000	38.964	41.630
MacOS[T.True]	8.2538	0.686	12.028	0.000	6.906	9.602
Windows[T.True]	3.9905	0.689	5.794	0.000	2.637	5.344
iOS[T.True]	-5.2566	0.704	-7.468	0.000	-6.640	-3.874
ActiveUsers	29.3976	1.251	23.499	0.000	26.940	31.856
ActiveUsers:MacOS[T.True]	8.5302	1.330	6.415	0.000	5.918	11.143
ActiveUsers:Windows[T.True]	-3.6770	1.301	-2.827	0.005	-6.233	-1.121
ActiveUsers:iOS[T.True]	-5.2718	1.347	-3.915	0.000	-7.918	-2.626
InteractingPct	18.8365	0.799	23.568	0.000	17.266	20.407
I(ActiveUsers ** 2)	-11.1165	0.940	-11.832	0.000	-12.962	-9.271
ActiveUsers:InteractingPct	-16.6230	1.546	-10.752	0.000	-19.661	-13.585

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.	11.9

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

Jako první otázku, kterou bych se zabýval, je to, jestli mám dostačující množství pozorování. Myslím si totiž, že by na "sociální síť" bylo potřeba výrazně více než 500 (bavil bych se asi v řádu mnoha tisíců, desetitisíců). Určitě by bylo vhodné brát data nějak periodicky a obecně velmi často, na mých datech není poznat, jestli nejsou brána náhodně. Na více testovacích datech bych také byl schopen na nějaké části regresní model natrénovat a na té další ho zvalidovat - tedy ověřit, že funguje správně. Také je v jednotlivých pozorováních určitě málo uživatelů, Facebook nebo Twitter (atd. nevím, co všechno se dneska používá) v jednom momentě používají statisíce nebo miliony lidí současně.

Co se týče mých statistik, tak z pohledu na **R-squared** je na tom model poměrně dobře (má zhruba `87.7%`). To znamená, že model velmi slušně fituje data (byť by to samozřejmě mohlo být o kousek lepší).

Významnost mého modelu taky docela podporuje můj výsledek **F-statistiky**, která je velmi nízko (`1.28e-215`), což je velmi dobrá hodnota.

Když se podívám na vyhodnocení parametrů, nejvíc model ovlivňuje množství aktivních uživatelů (`coef = 29.3976`) a procento interagujících uživatelů (`coef = 18.8365`), což napovídá, že primárně tyto parametry hrají hlavní roli při zvyšování odezvy. To i docela dává smysl, protože čím více uživatelů mám, tím víc mám zatížené servery nebo síť a tím pomaleji jsem schopný odpovídat.

Docela zajímavé je i se podívat na to, jaký vliv na ping mají různé operační systémy. Největší vliv má Windows (`coef = 3.9905`) a MacOS (`coef = 8.2538`), více však Windows. IOS má na ping spíše opačný efekt (`coef = -5.2566`).

Co se týče testů **Omnibus a Jarque-Berra**, tak u nich zde říct, že rezidua odpovídají normálnímu rozdělení, protože nejde zamítnout hypotézu, která u těchto testů říká, že *"rezidua odpovídají výběru z normálního rozdělení*.

Durbin-Wattsonův test (test autokorelace) je velmi blízko hodnotě 2 (`1.990`), což mi říká, že rezidua jsou mezi sebou nezávislá.

Ve zkratce bych to shrnul asi tak, že můj model vlastně není špatný a testy vycházejí velmi pěkně, nicméně bych měl trochu strach u malého počtu pozorování. Určitě by jich to chtělo více, abych dotával bezpečnější (nebo trochu důvěryhodnější výsledky).