MSP Druhý projekt

Autor: Maxim Plička, xplick04

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.stats import gamma, nbinom, truncnorm
import seaborn as sns
import os

import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from statsmodels.graphics.gofplots import qqplot
```

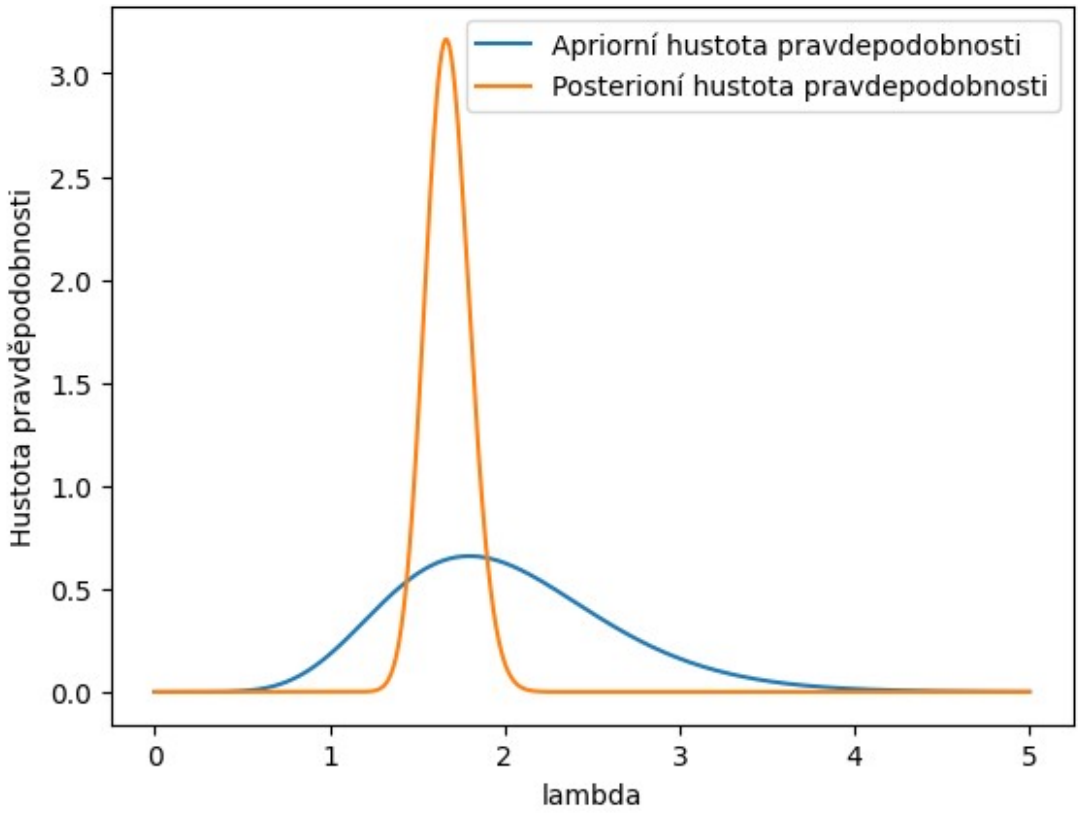
ÚLOHA 1

a)

1)

```
path = os.getcwd() + '/Projekt-2_Data.xlsx'
df = pd.read_excel(path)
df = df["uloha_1 a)"]


# Expertní odhad parametrů
lambda_e = 5
alpha_prior = 10
beta_prior = 1 / lambda_e
df_count = df.count()

# Posteriorní odhad parametrů
alpha_post = alpha_prior + df.sum()
beta_post = 1 / (lambda_e + df_count)

x = np.linspace(0, 5, 1000)

# Prior and posterior PDF
pdf_prior = gamma.pdf(x, alpha_prior, scale=beta_prior)
pdf_post = gamma.pdf(x, alpha_post, scale=beta_post)

# Plot PDF
```

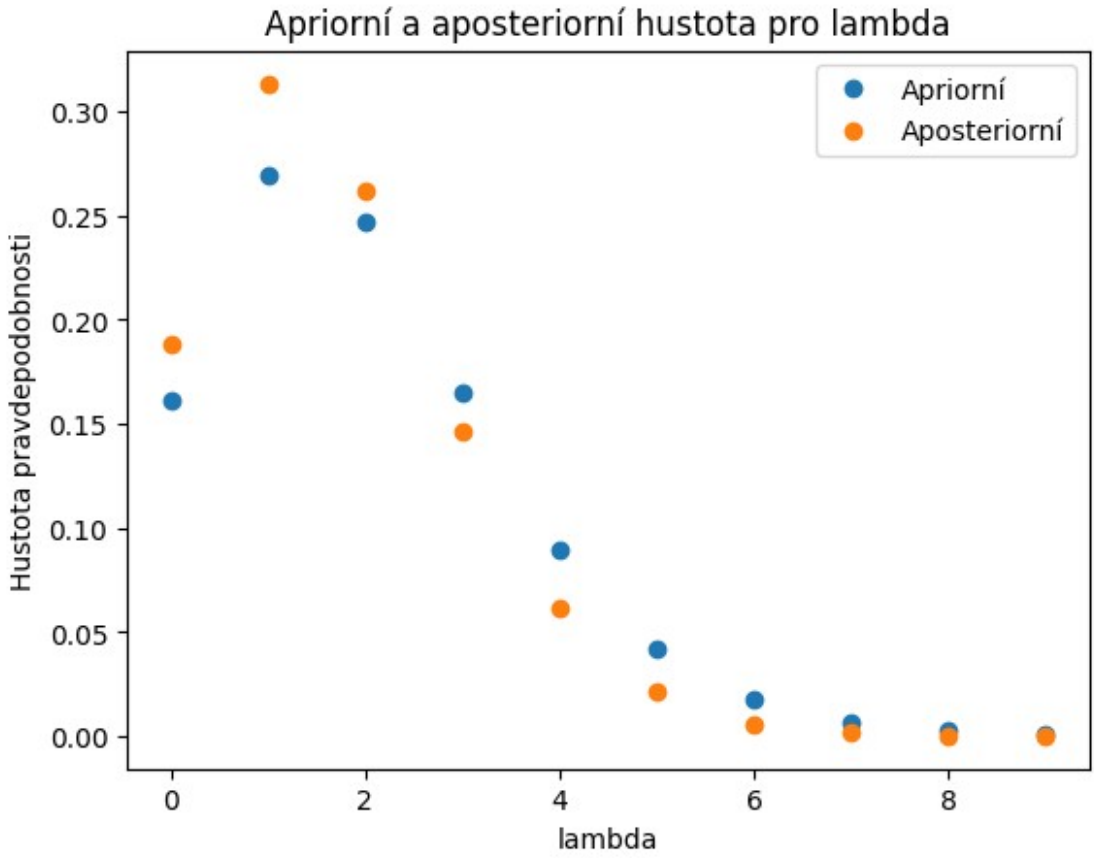
```
plt.plot(x, pdf_prior, label='Apriorní hustota pravdepodobnosti')
plt.plot(x, pdf_post, label='Posterioní hustota pravdepodobnosti')
plt.ylabel('Hustota pravděpodobnosti')
plt.xlabel('lambda')
plt.legend()
plt.show()
```



2)

```
x_values = np.arange(0, 10)

# Apriorní and posteriorní rozdělení
dist_prior = gamma(alpha_prior, loc=0, scale=beta_prior)
dist_post = gamma(alpha_post, loc=0, scale=beta_post)

# Apriorní and posteriorní hustota
prior_nbinom = nbinom.pmf(x_values, n=alpha_prior, p=lambda_e /
(lambda_e + 1) )
post_nbinom = nbinom.pmf(x_values, n=alpha_post, p=(lambda_e +
df_count) / (lambda_e + df_count + 1) )

plt.plot(x_values, prior_nbinom, 'o', label='Apriorní')
plt.plot(x_values, post_nbinom, 'o', label='Aposteriorní')
```

```
plt.title('Apriorní a aposteriorní hustota pro lambda')
plt.ylabel('Hustota pravdepodobnosti')
plt.xlabel('lambda')
plt.legend()
plt.show()
```



3)

```
print("95% Interval spolehlivosti pro lambdu z apriorního rozdělení:",
dist_prior.interval(0.95))
print("95% Interval spolehlivosti pro lambdu z aposteriorního
rozdělení:", dist_post.interval(0.95))
```

```
95% Interval spolehlivosti pro lambdu z apriorního rozdělení:
(0.9590777392264868, 3.416960690283833)
95% Interval spolehlivosti pro lambdu z aposteriorního rozdělení:
(1.4376938284869922, 1.9327207471868797)
```

4)

```
median_posterior = gamma.ppf(0.5, alpha_post, scale=beta_post)
mean_posterior = gamma.mean(alpha_post, scale=beta_post)
```

```
print("Medián:", median_posterior)
print("Střední hodnota:", mean_posterior)
```

```
Medián: 1.6730169441241727
Střední hodnota: 1.6761904761904765
```

Vybral jsem si střední hodnotu a medián, jelikož se jedná o jedny z nejzákladnějších odhadů.

Hodnoty na obou stranách střední hodnoty jsou přibližně stejně vzdáleny od mediánu, což značí že data mají symetrické rozdělení.

5)

```
med_prior = nbinom.median(n=alpha_prior, p=lambda_e / (lambda_e + 1) )
mean_post = nbinom.median(n=alpha_post, p=(lambda_e + df_count) /
(lambda_e + df_count + 1) )

print("Medián apriorního odhadu:", med_prior)
print("Madián aposteriorního odhadu:", mean_post)
```

```
Medián apriorního odhadu: 2.0
Madián aposteriorního odhadu: 1.0
```

Vybral jsem mediány apriorního a aposteriorního rozložení.

Madiány rozložení se lehce liší, což může značit, že nová data způsobila změnu střední hodnoty. Navíc se i lehce změnila poloha rozožení.

ÚLOHA 1

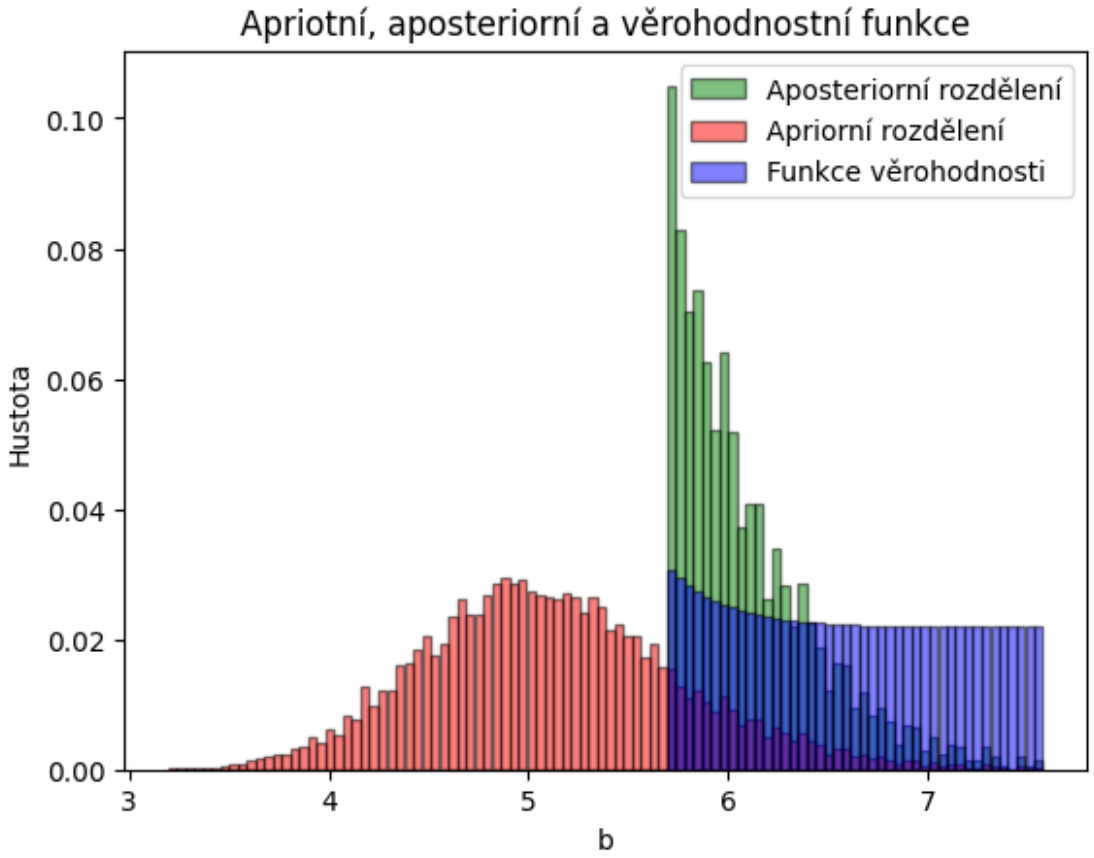
b)

1)

```
path = os.getcwd() + '/Projekt-2_Data.xlsx'
df = pd.read_excel(path)


df_pozorovani = df["uloha_1 b)_pozorování"].dropna()
df_1 = df[["uloha_1 b)_prior", "skupina"]]
df_1 = df_1.groupby(["skupina"], observed=True).max().reset_index()
# maxima pro kazdou skupinu
df_1 = df_1["uloha_1 b)_prior"]

# apirorní funkce
zakosikovana_data = pd.cut(df_1, bins=100)
stredy = zakosikovana_data.unique().categories.mid
```

```
pravdepodobnosti = df_1.groupby(zakosikovana_data,
observed=False).count() / len(df_1)
pravdepodobnosti = pravdepodobnosti.reset_index(drop=True)
df_apriori = pd.DataFrame({"b": stredy, "h(b)": pravdepodobnosti})


# likehood funkce
# trunkované normální rozdělení parametry
mean = 3
std = 1
a = 1

df_likehood = pd.DataFrame({"b": stredy, "p(b)" : 0})
for b in stredy:
    distribution_t = truncnorm((a - mean) / std, (b - mean) / std,
loc=mean, scale=std)
    pb = 1
    for p in df_pozorovani:
        pb = pb * distribution_t.pdf(p)
    df_likehood.loc[df_likehood["b"] == b, "p(b)"] = float(pb)
    df_likehood["p(b)"] = df_likehood["p(b)"].astype(float)

df_likehood["p(b)"] = df_likehood["p(b)"] / df_likehood["p(b)"].sum()


# aposteriorní funkce
df_aposteriori = df_likehood.copy()
df_aposteriori.drop(columns=["p(b)"], inplace=True)
df_aposteriori['p(b|h)'] = df_likehood['p(b)'] * df_apriori['h(b)']


# normalizace
suma = 0
for b in stredy:
    suma = suma + df_apriori['h(b)'].loc[df_apriori["b"] ==
b].values[0] * df_likehood['p(b)'].loc[df_likehood["b"] ==
b].values[0]

df_aposteriori['p(b|h)'] = df_aposteriori['p(b|h)'] / suma


plt.bar(df_aposteriori['b'], df_aposteriori['p(b|h)'], width=0.043,
alpha=0.5, color='g', edgecolor='black')
plt.bar(df_apriori['b'], df_apriori['h(b)'], width=0.043, alpha=0.5,
color='r', edgecolor='black')
plt.bar(df_likehood['b'], df_likehood['p(b)'], width=0.043, alpha=0.5,
color='b', edgecolor='black')
plt.xlabel('b')
plt.ylabel('Hustota')
plt.title('Apriotní, aposteriorní a věrohodnostní funkce')
```

```
plt.legend(['Aposteriorní rozdělení', 'Apriorní rozdělení', 'Funkce věrohodnosti'])
plt.show()
```



2)

```
# 95% interval spolehlivosti
sorted_b_values = np.sort(df_aposteriori['b'])
cdf_values = np.cumsum(df_aposteriori['p(b|h)'])
lower_index = np.argmax(cdf_values >= 0.025)
upper_index = np.argmax(cdf_values >= 0.975)

# hodnoty b na hranicích intervalu
lower_bound = sorted_b_values[lower_index]
upper_bound = sorted_b_values[upper_index]


print(f"95% Interval spolehlivosti pro b z aposteriorního rozložení: ({lower_bound}, {upper_bound}) ")
```

```
95% Interval spolehlivosti pro b z aposteriorního rozložení: (5.716, 7.031000000000001)
```

3)

```
# výpočet střední hodnoty a mediánu
mean = np.sum(df_aposteriori['b'] * df_aposteriori['p(b|h)'])
df_aposteriori = df_aposteriori.sort_values('b')
median =
df_aposteriori['b'].iloc[np.argmax(np.cumsum(df_aposteriori['p(b|h)'])
>= 0.5)]

print(f"Střední hodnota (očekávaná hodnota) parametru b: {mean}")
print(f"Medián parametru b: {median}")
```

```
Střední hodnota (očekávaná hodnota) parametru b: 6.095022322519652
Medián parametru b: 5.979
```

Úloha 2

1)

```
def backward_elimination(formula, data):
    while True:
        model = smf.ols(formula, data=data).fit()
        p_values = model.pvalues.drop('Intercept')
        max_p_value = p_values.max()

        if max_p_value > 0.05:
            # Odstranění proměnné s nejvyšší p-hodnotou
            variable_to_remove = p_values.idxmax()
            print(variable_to_remove)
            formula = formula.replace(f' + {variable_to_remove}', '')
            formula = formula.replace(f' {variable_to_remove}', '')
        else:
            break
    return formula

def standartization(data, columns):
    for column in columns:
        data[column] = (data[column] - data[column].mean()) /
data[column].std()
        #druhý způsob standardizace
        #data_copy[column] = (data_copy[column] -
((data_copy[column].max() + data_copy[column].min()) / 2)) /
((data_copy[column].max() - data_copy[column].min()) / 2)
    return data


# Načtení dat
path = os.getcwd() + '/Projekt-2_Data.xlsx'
```

```
df = pd.read_excel(path, sheet_name="Úloha 2")

# Vytvoření one-hot encoding pro 'OSType', pouze 3/4 hodnot,
multikolinearita
df['OS_Windows'] = (df['OSType'] == 'Windows').astype(int)
df['OS_IOS'] = (df['OSType'] == 'iOS').astype(int)
df['OS_MacOS'] = (df['OSType'] == 'MacOS').astype(int)

# Pokud jsou všechny ostaní hodnoty OS nulové nulové, tak je to
zbytečný sloupec
df.drop('OSType', axis=1, inplace=True)
# InteractingPct = 1 - ScrollingPct, multikolinearita
df.drop('InteractingPct', axis=1, inplace=True)

new_column_names = {'Ping [ms]': 'Ping', 'ActiveUsers': 'AU',
'ScrollingPct': 'SP', 'OS_Windows': 'os_w', 'OS_IOS': 'os_i',
'OS_MacOS': 'os_m'}
df.rename(columns=new_column_names, inplace=True)

# Standardizace
df_og = df.copy()
df = standartization(df, ['AU', 'SP'])

# nemá smysl dávat OSType dohromady a ani na druhou, odstranění I(AU
** 2) z důvodu velké VIF (větší než 10)
formula_in = 'Ping ~ AU + SP + os_w + os_i + os_m + AU:SP + AU:os_w +
AU:os_i + AU:os_m  + SP:os_w + SP:os_i + SP:os_m + I(SP ** 2)'

print("Kofeciety odstraněné během zpětné eliminace:")
formula = backward_elimination(formula_in, df)
model = smf.ols(formula, data=df).fit()

# Výpis výsledků
intercept = model.params['Intercept']
coefficients = model.params.drop('Intercept')
variable_names = coefficients.index
equation = f'Ping = {intercept:.4f} + ' + ' +
'.join([f'{coeff:.4f}*{var}' for coeff, var in zip(coefficients,
variable_names)])


print(model.summary())
print("Regresní rovnice:")
print(equation)
```

```
Kofeciety odstraněné během zpětné eliminace:
SP:os_w
SP:os_i
SP:os_m
I(SP ** 2)
```

```
AU:os_w
AU:os_i
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                   Ping   R-squared:
0.813
Model:                            OLS   Adj. R-squared:
0.810
Method:                 Least Squares   F-statistic:
306.0
Date:                Sun, 17 Dec 2023   Prob (F-statistic):
4.22e-175
Time:                        20:32:11   Log-Likelihood:
-1644.2
No. Observations:                 502   AIC:
3304.
Df Residuals:                     494   BIC:
3338.
Df Model:                           7

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
Intercept     48.6769      0.611     79.659      0.000      47.476
49.878
AU             8.3421      0.339     24.597      0.000       7.676
9.008
SP            -5.1143      0.290    -17.632      0.000      -5.684
-4.544
os_w           3.3925      0.827      4.103      0.000       1.768
5.017
os_i          -6.1473      0.855     -7.190      0.000      -7.827
-4.468
os_m           9.1153      0.823     11.075      0.000       7.498
10.732
AU:SP          2.4170      0.293      8.251      0.000       1.841
2.992
AU:os_m        5.5536      0.656      8.462      0.000       4.264
6.843
==============================================================================
========
Omnibus:                      113.115   Durbin-Watson:
1.872
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
584.177
Skew:                           0.876   Prob(JB):
1.40e-127
Kurtosis:                       7.986   Cond. No.
5.08
================================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
Regresní rovnice:
Ping = 48.6769 + 8.3421*AU + -5.1143*SP + 3.3925*os_w + -6.1473*os_i +
9.1153*os_m + 2.4170*AU:SP + 5.5536*AU:os_m
```

Základní předpoklady lineární regrese a regresní diagnostiky:

Lineární vztah mezi nezávislými a závislou proměnnou: Z grafu Rezidua vs Predikované hodnoty, lze vidět, že data jsou náhodně rozložena kolem osy v nule. To implikuje, že lineární vztah mezi nezávislými prediktory a závislou proměnou je splňen.

Multikolinearita: Hodnoty VIF ukazují, že multikolinearita je minimální. Ani jedna z hodnot nepřesáhla práh 10. K tomuto výsledku bylo z původní formule modelu potřeba odstranit proměnou ActiveUsers ** 2, která narušovala tento předpoklad.

Homoskedasticita: Z grafu "Rezidua vs Pořadí" a "Rezidua vs Predikované hodnoty" lze vidět, že se rozložení reziduí kolem osy je relativně stejné (objevují se jen lechce méně zahuštěné části).

Normální rozdělení reziduí: Z histogramu reziduí lze vyčíst, že se velice blíží zvonovému tvaru normálního rozložení. Navíc se také rezidua v QQ grafu blíží k přímce.

Nezávislost reziduí: Hodnota durbin-Watsonova testu se blíží 2 (konkrétně 1,872), což značí nezávislost reziduí

Odlehlé hodnoty: Z grafů níže lze vyčíst, že se mezi rezidui vyskytují některé odlehlé hodnoty, které níže pomocí podmínek z democvičení odstraňuji. K jejich odstranění se konkrétně využívá Leverage, Cookova vzdálenost a vzdálenosti standardizovaných reziduí. Tyto hodnoty jsou odstraněny níže, jelikož mohou negativně ovlivnit učení modelu.

```
def plot_residual_histogram(residuals):
    plt.hist(residuals, bins='auto', density=True, alpha=0.6,
color='g')

    # Gaussova křivka
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = stats.norm.pdf(x, np.mean(residuals), np.std(residuals))
```

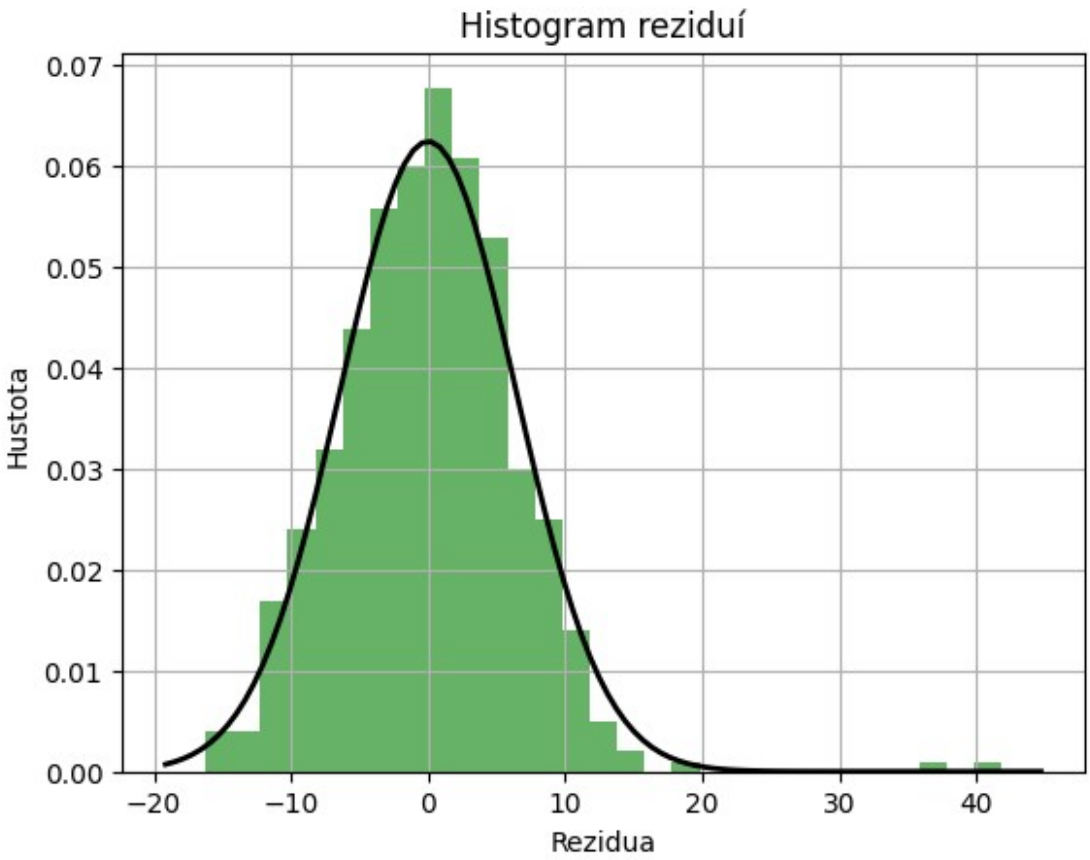
```
    plt.plot(x, p, 'k', linewidth=2)

    plt.grid(True)
    plt.title("Histogram reziduí")
    plt.xlabel("Rezidua")
    plt.ylabel("Hustota")
    plt.show()

# Q-Q graf
def plot_qq_plot(residuals):
    qqplot(residuals, line='s')
    plt.title('Q-Q graf')
    plt.grid(True)
    plt.show()

# Rezidua vs. predikované hodnoty
def plot_res_vs_fit(fitted_values, residuals):
    plt.scatter(fitted_values, residuals)
    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()

# Rezidua vs. pořadí
def plot_res_vs_order(residuals):
    plt.scatter(range(len(residuals)), residuals, alpha=0.5)
    plt.axhline(y=0, color='r', linestyle='-')
    plt.grid(True)
    plt.title("Rezidua vs Pořadí")
    plt.xlabel("Pořadí")
    plt.ylabel("Rezidua")
    plt.show()

def calculate_vif(formula, data):
    model = smf.ols(formula, data=data)
    X = pd.DataFrame(model.exog, columns=model.exog_names)

    # Výpočet VIF pro každý prediktor
    vif = pd.Series([variance_inflation_factor(X.values, i) for i in
range(X.shape[1])], index=X.columns)
    # Vytvoření DataFrame s VIF
    vif_df = vif.to_frame()
    vif_df.columns = ['VIF']  # Nastavení názvu sloupce
    print(vif_df)
    # Korelace prediktorů
    print('\n')
    print(X.corr())
```

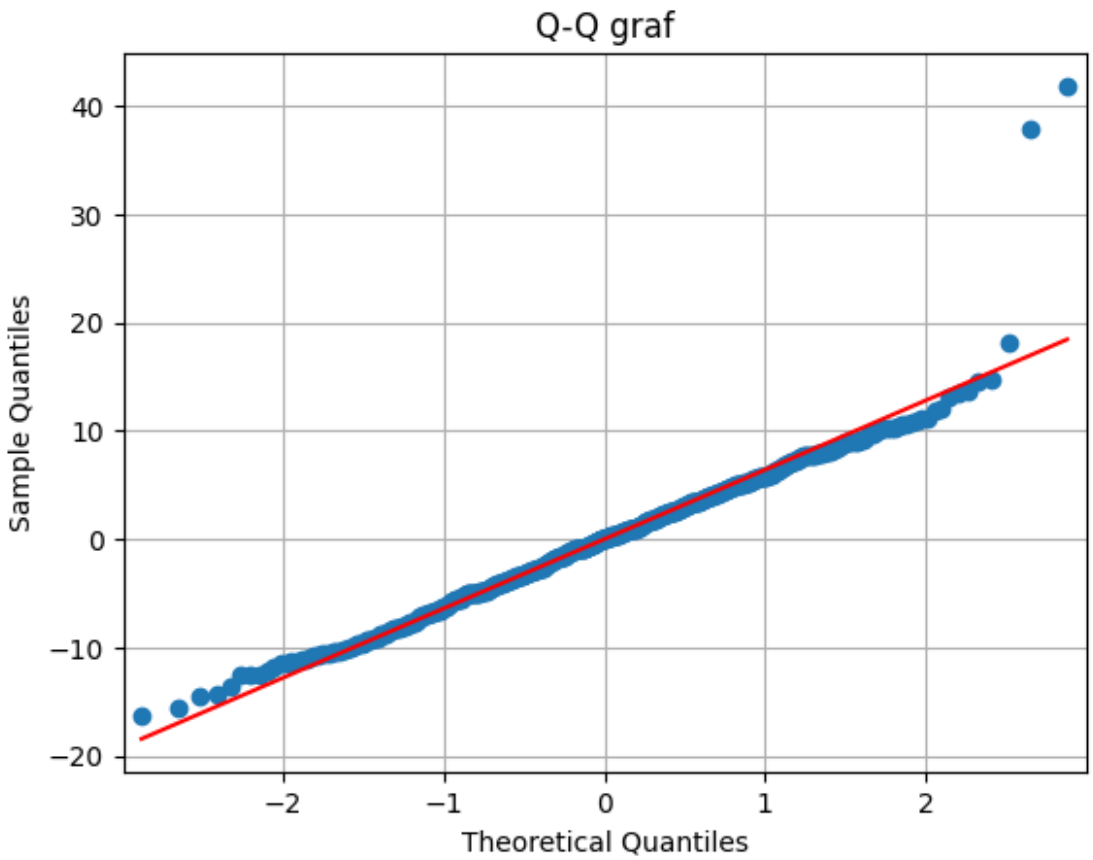
```
# Výpočet reziduí (pracuje se s originálním datasetem, ne
standardizovaným)
model = smf.ols(formula, data=df_og).fit()


calculate_vif(formula, df_og)
plot_residual_histogram(model.resid)
plot_qq_plot(model.resid)
plot_res_vs_fit(model.fittedvalues, model.resid)
plot_res_vs_order(model.resid)
```

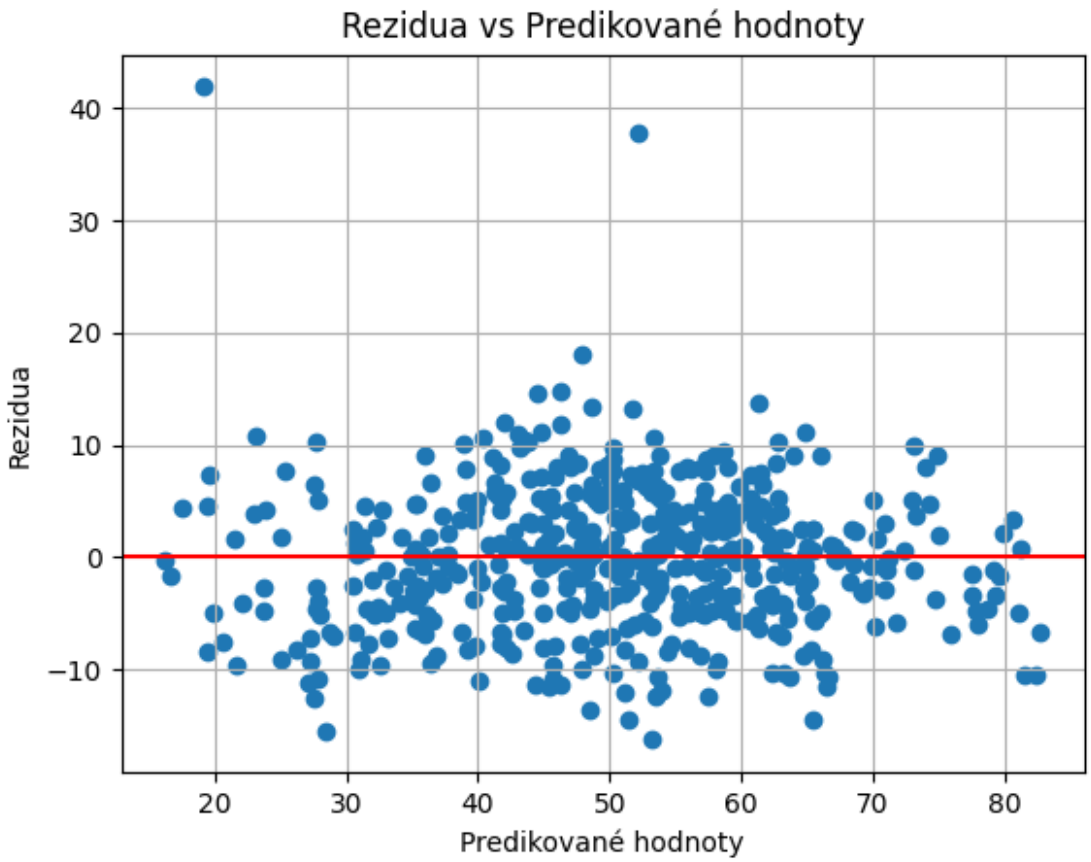
```
                 VIF
Intercept  30.774141
AU          4.769242
SP          5.630743
os_w        1.613865
os_i        1.585129
os_m        6.581920
AU:SP       8.477600
AU:os_m     6.175504
```

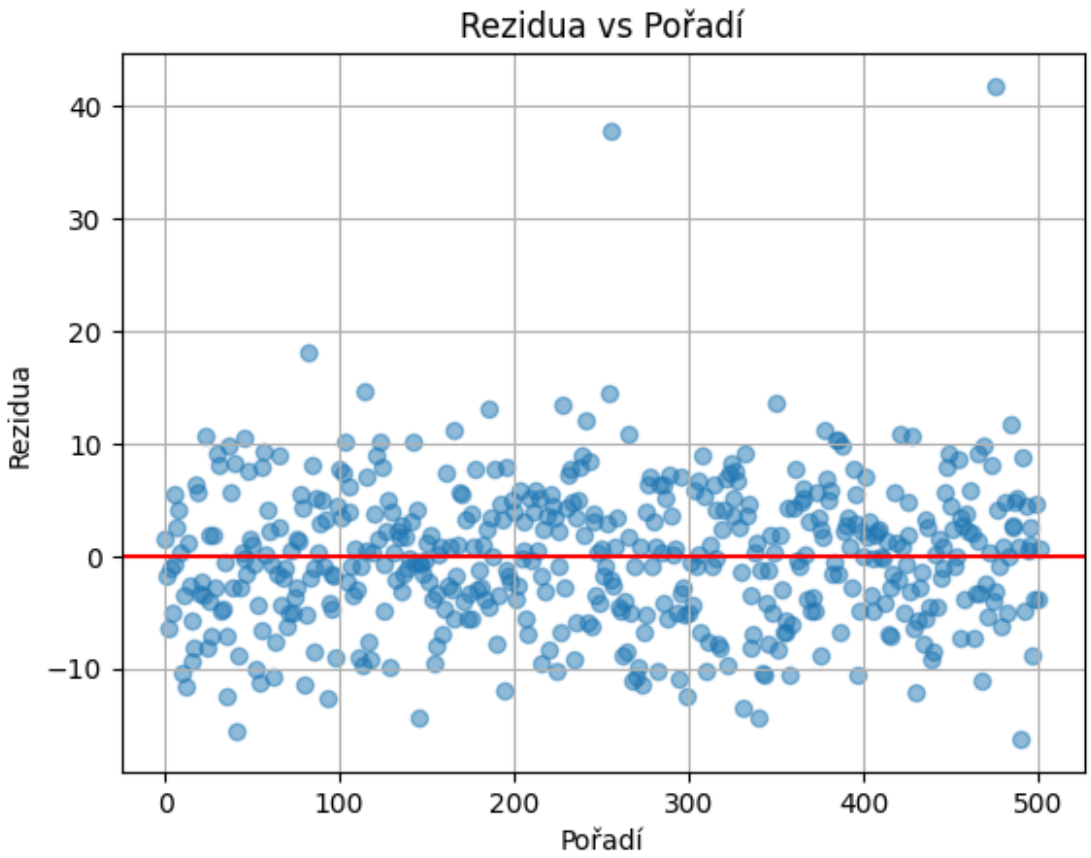
```
           Intercept        AU        SP      os_w      os_i      os_m
\
Intercept        NaN       NaN       NaN       NaN       NaN       NaN

AU               NaN  1.000000 -0.040275  0.003135 -0.063206 -0.000136

SP               NaN -0.040275  1.000000  0.016964  0.062634 -0.086466

os_w             NaN  0.003135  0.016964  1.000000 -0.334506 -0.371550

os_i             NaN -0.063206  0.062634 -0.334506  1.000000 -0.341322

os_m             NaN -0.000136 -0.086466 -0.371550 -0.341322  1.000000

AU:SP            NaN  0.582505  0.711749  0.001067  0.011121 -0.053109

AU:os_m          NaN  0.243697 -0.072359 -0.327426 -0.300788  0.881244
```

```
              AU:SP   AU:os_m
Intercept       NaN       NaN
AU         0.582505  0.243697
SP         0.711749 -0.072359
os_w       0.001067 -0.327426
os_i       0.011121 -0.300788
os_m      -0.053109  0.881244
AU:SP      1.000000  0.091981
AU:os_m    0.091981  1.000000
```


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


Q-Q graf
Sample Quantiles
Theoretical Quantiles
40
30
20
10
0
−10
−20
−2
−1
0
1
2


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

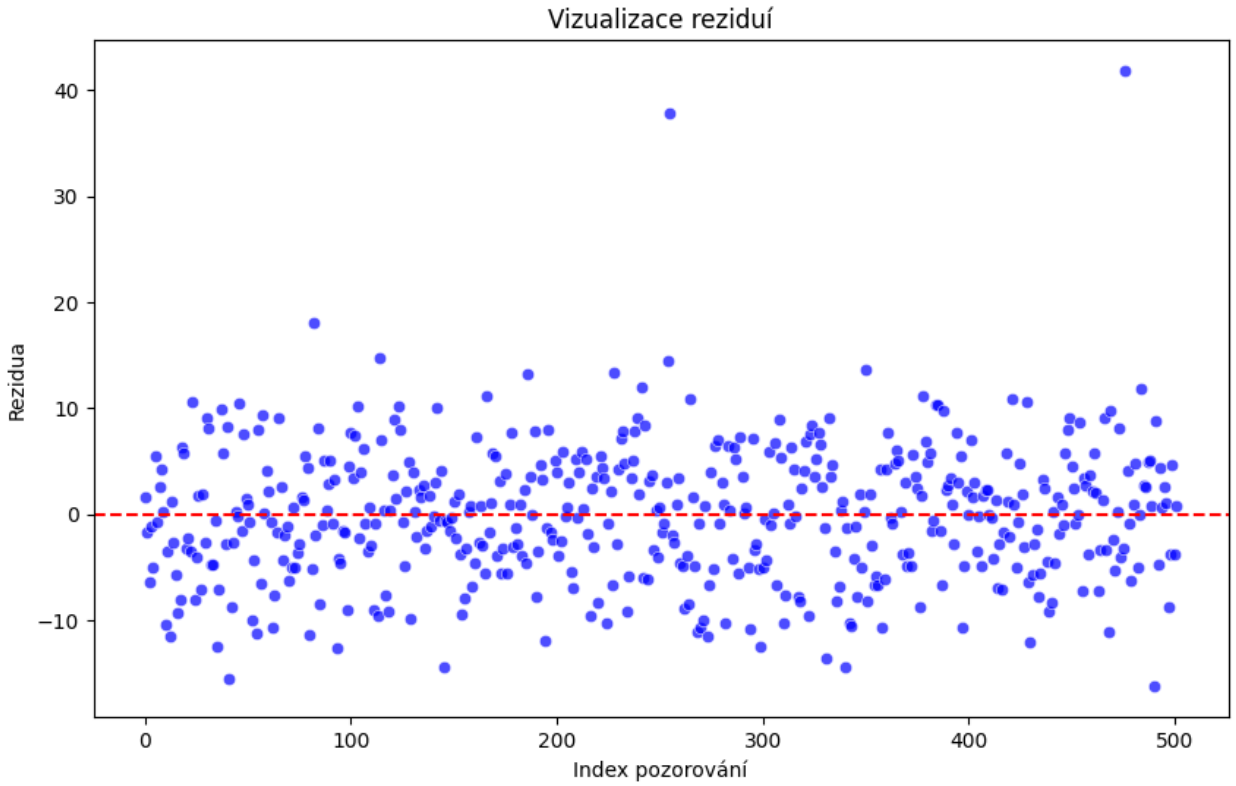


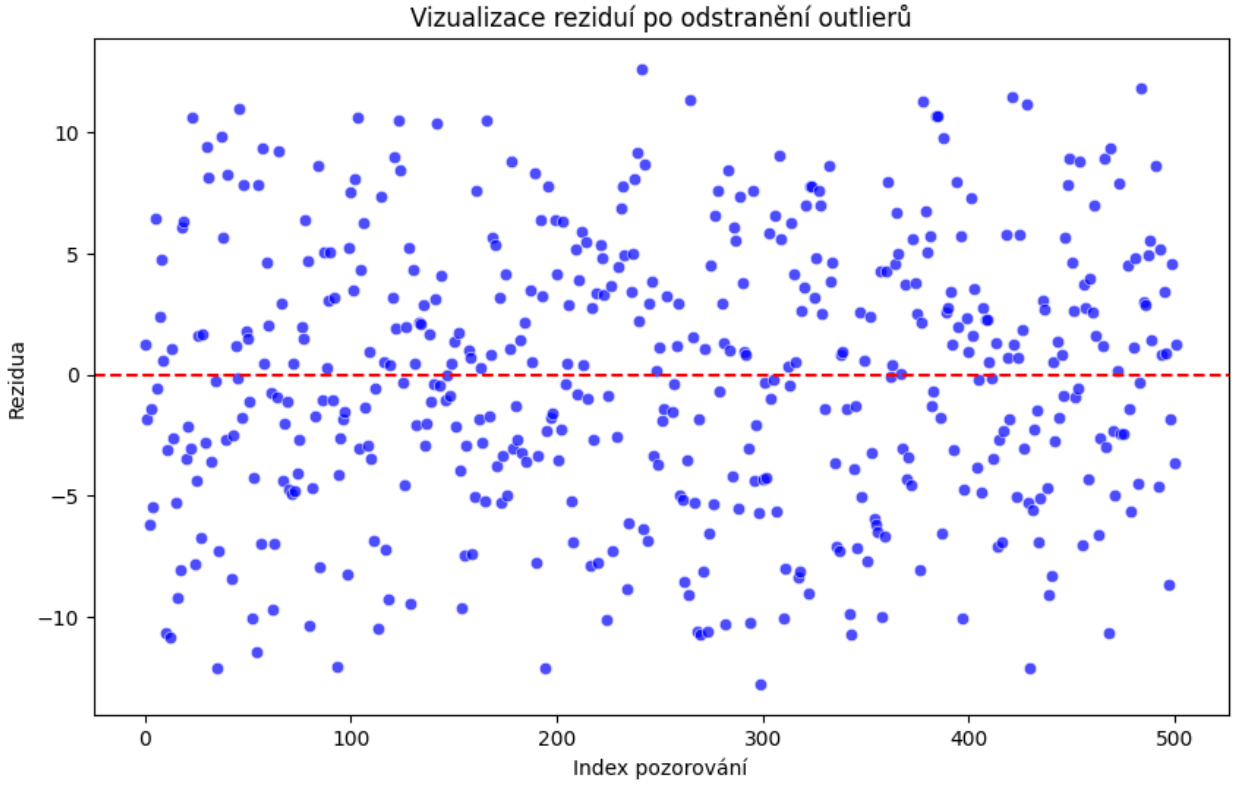
Identifikace odlehlých hodnot

Hodnoty je možné odstranit, jelikož negativně ovlivňují učící faktor modelu.

```
# Výpočet reziduí
model = smf.ols(formula, data=df_og).fit()
residuals = model.resid
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df_og.index, y=residuals, color='blue', alpha=0.7)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Index pozorování')
plt.ylabel('Rezidua')
plt.title('Vizualizace reziduí')
plt.show()


influence = model.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
# Vytvoření DataFrame s výsledky
outl_stats_df = pd.DataFrame({
    'Leverage': leverage,
    'Standardized Residuals': standardized_residuals,
    'Studentized Residuals': studentized_residuals,
    'Cook\'s Distance': cooks_d[0],
    'Cook\'s Distance_p-value': cooks_d[1]
}, index=df_og.index)

merged_df = df_og.join(outl_stats_df)


#Odstranění outlierů (příliš velké leverage, příliš velké Cookovy D
hodnoty, příliš velké standardizované rezidua)
outl_stats_df = merged_df[(outl_stats_df['Leverage'] <=
3*len(model.params)/df_og.shape[0]) &
(np.abs(outl_stats_df['Standardized Residuals']) <= 2) &
(outl_stats_df['Cook\'s Distance_p-value'] >= 0.05)]

model = smf.ols(formula, data=outl_stats_df).fit()
residuals = model.resid
plt.figure(figsize=(10, 6))
sns.scatterplot(x=outl_stats_df.index, y=residuals, color='blue',
alpha=0.7)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Index pozorování')
plt.ylabel('Rezidua')
plt.title('Vizualizace reziduí po odstranění outlierů')
plt.show()
```


Vizualizace reziduí
Reziduа
Index pozorování
40
30
20
10
0
−10
0
100
200
300
400
500


Vizualizace reziduí po odstranění outlierů
Reziduа
Index pozorování
10
5
0
−5
−10
0
100
200
300
400
500

2)

```
# Výpočet predikce
index = model.predict().argmax()

print(f"Největší predikovaná hodnota: {model.predict()[index]:.4f}")
```

```
Největší predikovaná hodnota: 83.2686
```

Pro identifikaci nastavení parametrů, které způsobí nejproblematičtější odezvu je potřeba zjistit, jaký mají jednotlivé proměnné vliv na hodnotu odezvy.

Mezi koeficienty zvyšující odezvu patří ActiveUsers, windows, macOS a interakční termíny ActiveUsers * ScrollingPct a ActiveUsers * macOS.

Mezi koeficienty snižující odezvu patří ScrollingPct a IOS.

V případě nalezení nejproblematičtější hodnoty je třeba najít kombinaci s vysokými hodnotami, které zvyšují odezvu a s nízkými nebo nulovými hodnotami, které sniřují odezvu.

3)

```
# Průměrné hodnoty pro standardizované proměnné AU a IP
mean_AU = df['AU'].mean()
mean_IP = df['SP'].mean()

# Vytvoření nového datového rámce pro predikci s průměrnými hodnotami proměnných AU a IP
new_data = pd.DataFrame({
    'AU': [mean_AU],
    'SP': [mean_IP],
    'os_w': [1],
    'os_i': [0],
    'os_m': [0],
    'AU:SP': [mean_AU * mean_IP],
    'AU:os_w': [mean_AU * 1],
    'AU:os_i': [mean_AU * 0],
    'AU:os_m': [mean_AU * 0],
    'SP:os_w': [mean_IP * 1],
    'SP:os_i': [mean_IP * 0],
    'SP:os_m': [mean_IP * 0],

})


prediction = model.get_prediction(new_data)
pred = prediction.summary_frame(alpha=0.05)  # 95% intervaly

print("Odhadnutá hodnota odezvy uživatele s Windows:")
print(f'Predikce: {pred["mean"].values[0]:.4f}')
print(f'Predikční interval: ({pred["mean_ci_lower"].values[0]:.4f},
```

```
{pred["mean_ci_upper"].values[0]:.4f})')
print(f'Konfidenční interval: ({pred["obs_ci_lower"].values[0]:.4f},
{pred["obs_ci_upper"].values[0]:.4f})')
```

```
Odhadnutá hodnota odezvy uživatele s Windows:
Predikce: 52.9116
Predikční interval: (50.2010, 55.6223)
Konfidenční interval: (41.6582, 64.1651)
```

4)

Model má schopnost dobře vysvětlit variabilitu závislé proměnné "ping", což lze vyčíst z hodnoty vysoké hodnoty R-squared (konkrétně 0.813).

Honodta upraveného R-squared, která zohledňuje počet proměnných v modelu a mění se dle jejich přínosu (pokud je jejich přinos pro model lepší/horší než očekávaný přínos). Tato hodnota je vysoká (0.81), coz značí, že nezávislé proměnné jsou zvoleny dobře. Navíc jsou všeschny tyto hodnoty statisticky významné, což naznačuje hodnota p-value v přehledu modelu (všechny nevýznamné hodnoty byly odstraněny při zpětné eliminaci). Z odpovědi na otázku 2 plyne, že model splňuje regresní předpoklady.

Ze všech těchto pozorování plyne, že model je vhodný pro další použití. Bylo by ovšem vhodné tento model prvně validovat na nějakém validačním datasetu, aby se ověřila jeho predikční schopnost.