jahn-msp-2-projekt

December 17, 2023

1 MSP 2. projekt

1.1 Import knihoven

[1]:
```
import pandas as pd
import numpy as np
import scipy.stats as st
import scipy.special as sp
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.stats.outliers_influence as smso
import statsmodels.graphics.gofplots as splt
import IPython.display as id
from scipy.stats import truncnorm
from scipy.special import softmax
from statsmodels.stats.outliers_influence import variance_inflation_factor

dataSet1 = pd.read_excel("Projekt-2_Data.xlsx", sheet_name="Úloha 1")
dataSet2 = pd.read_excel("Projekt-2_Data.xlsx", sheet_name="Úloha 2")
```

2 1. úloha – Bayesovské odhady

2.1 a) Konjugované apriorní a aposteriorní rozdělení, prediktivní rozdělení

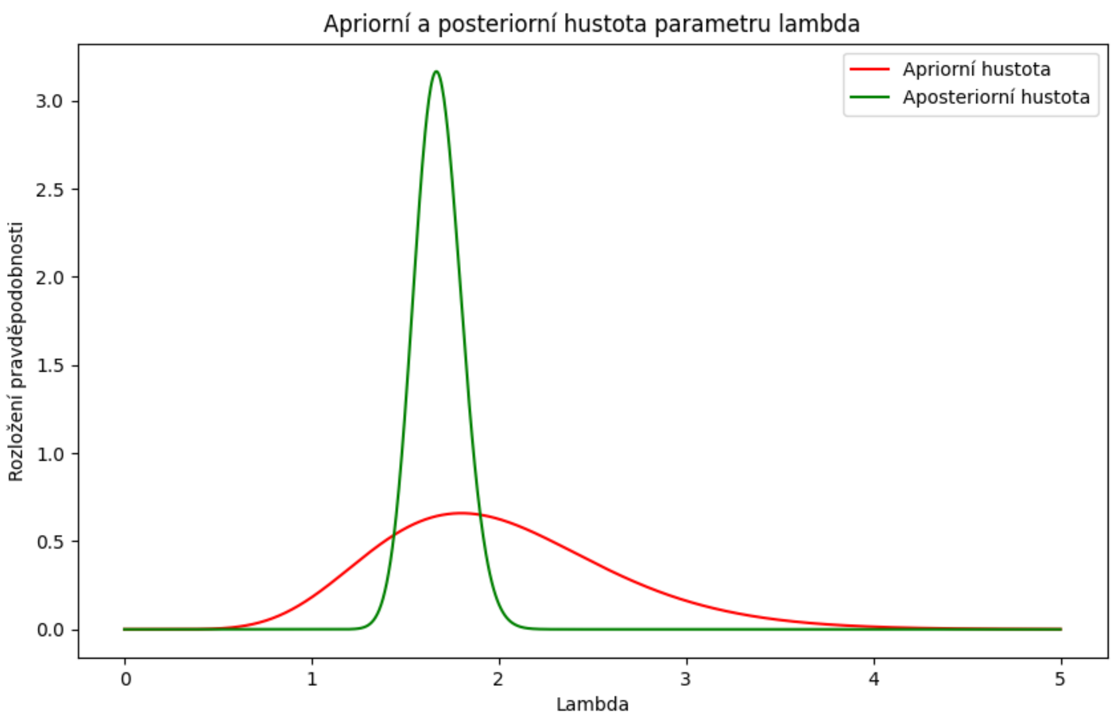
Expertní odhad pro náhodnou veličinu s Poissonovým rozdělením naznačuje, že by mělo dojít k 10 připojením (celkem 10 událostí) v průběhu každých 5 ms (5 časových intervalů). Příslušné apriorní konjugované rozdělení je popsáno Gamma rozdělením s parametry $\alpha = 10$ a $\beta = 5$.

2.1.1 1) Apriorní a aposteriorní hustotou parametru Poissonova rozdělení

Apriorní hustota je pro nás, jak jsme si již řekli, Gama rozdělení. Aposteriorní hustota je pak hustota Gama rozdělení s parametry viz tabulka https://en.wikipedia.org/wiki/Conjugate_prior

[2]:
```
# apriorni parametry
alphaPrior = 10
betaPrior = 5

# vyfiltrovana data
```

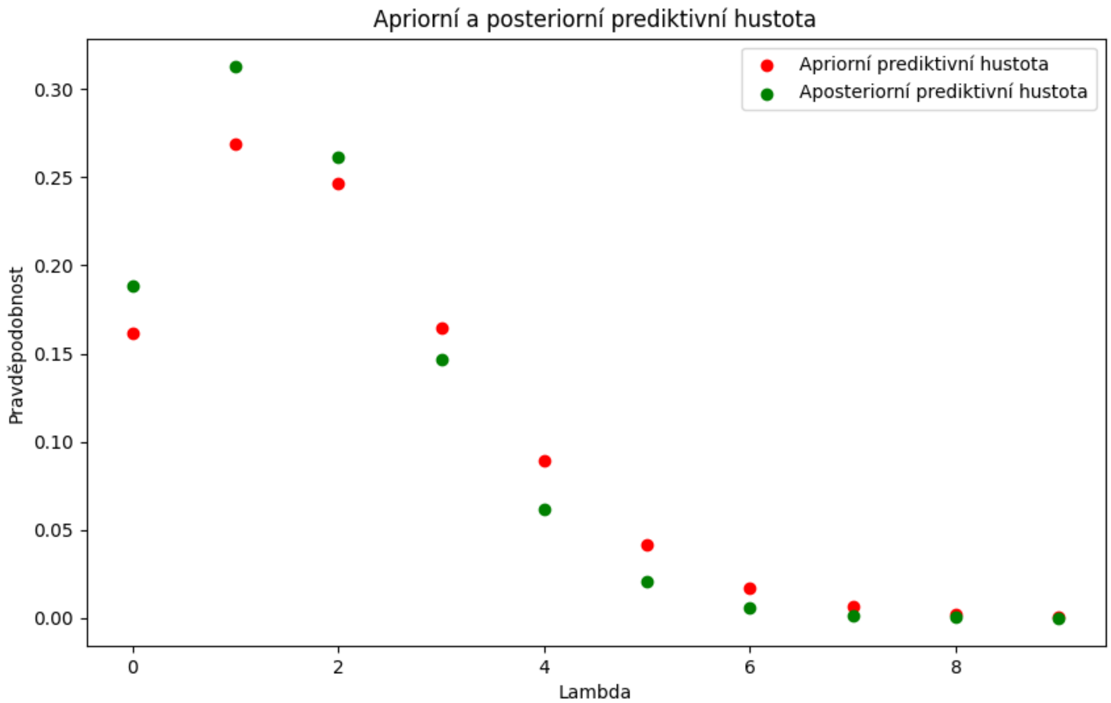
```python
observations = np.array(dataSet1["uloha_1 a)"].dropna().values)

# aposteriorni parametry
alphaPost = alphaPrior + observations.sum()
betaPost = betaPrior + len(observations)

# lambda - 1000 bodu mezi 0 a max
lambdas = np.linspace(0, observations.max(), 1000)

# pravdepodobnostni funkce
prior = st.gamma.pdf(lambdas, alphaPrior, scale=1/betaPrior)
posterior = st.gamma.pdf(lambdas, alphaPost, scale=1/betaPost)

# graf
plt.figure(figsize=(10, 6))
plt.plot(lambdas, prior, label='Apriorní hustota', color='red')
plt.plot(lambdas, posterior, label='Aposteriorní hustota', color='green')
plt.legend()
plt.title('Apriorní a posteriorní hustota parametru lambda')
plt.xlabel('Lambda')
plt.ylabel('Rozložení pravděpodobnosti')
plt.show()
```



2.1.2 2) Apriorní a aposteriorní prediktivní hustotou pozorovaní za jeden časový interval

Pro výpočet parametru prediktivní hustoty využijeme binomického rozdělení, jehož apriorní parametry jsou rovněž uvedeny v tabulce.

[3]:
```
# apriorni parametry
binomialAlphaApriori = alphaPrior
binomialBetaPrior = betaPrior/(betaPrior + 1)

# posteriorni parametry
binomialAlphaPost = alphaPost
binomialBetaPost = betaPost/(betaPost + 1)

k = np.arange(0, 10) # experimentalne nastaveno na 10

predictive_probs_prior = st.nbinom.pmf(k, binomialAlphaApriori,␣
 ↪binomialBetaPrior)
predictive_probs_posterior = st.nbinom.pmf(k, binomialAlphaPost,␣
 ↪binomialBetaPost)

plt.figure(figsize=(10, 6))
plt.scatter(k, predictive_probs_prior, label='Apriorní prediktivní hustota',␣
 ↪color='red')
plt.scatter(k, predictive_probs_posterior, label='Aposteriorní prediktivní␣
 ↪hustota', color='green')
plt.title('Apriorní a posteriorní prediktivní hustota')
plt.xlabel('Lambda')
plt.ylabel('Pravděpodobnost')
plt.legend()
plt.show()
```



2.1.3 3) 95% interval spolehlivosti pro parametr z apriorního a aposteriorního rozdělení

Interval spolehlivosti odhadu parametru λ pro požadovanou hladinu spolehlivosti 95 % je dán výrazem $(\hat{\lambda} g_{1-\alpha/2}, \hat{\lambda} g_{\alpha/2})$, kde:

- $\hat{\lambda}$ představuje bodový odhad parametru λ,
- $1-\alpha$ určuje hladinu významnosti (pro 95 % spolehlivost je $\alpha = 0.05$),
- g_k značí k-tý percentil Gamma rozdělení.

Obě apriorní i aposteriorní rozdělení jsou modelována pomocí Gamma rozdělení, což znamená, že bodové odhady jejich parametrů λ jsou definovány následovně:

$$\hat{\lambda} = \frac{\alpha + \sum_{i=1}^{n} x_i}{\beta + n}.$$

Konečně můžeme vypočítat, že interval je ohraničen na [2.5; 97.5] percentil Gamma rozdělení.

[4]:
```
# interval spolehlivosti pro parametr lambda z apriorního a aposteriorního␣
↪rozdělení
lambdaPrior025 = st.gamma.ppf(0.025, a=alphaPrior, scale=1/betaPrior)
lambdaPrior975 = st.gamma.ppf(0.975, a=alphaPrior, scale=1/betaPrior)

lambdaPost025 = st.gamma.ppf(0.025, a=alphaPost, scale=1/betaPost)
lambdaPost975 = st.gamma.ppf(0.975, a=alphaPost, scale=1/betaPost)
```

```
print(f'95% interval spolehlivosti pro parametr lambda z apriorního rozdělení:␣
↪<{round(lambdaPrior025, 5)}, {round(lambdaPrior975, 5)}>')
print(f'95% interval spolehlivosti pro parametr lambda z aposteriorního␣
↪rozdělení: <{round(lambdaPost025, 5)}, {round(lambdaPost975, 5)}>')
```

```
95% interval spolehlivosti pro parametr lambda z apriorního rozdělení: <0.95908,
3.41696>
95% interval spolehlivosti pro parametr lambda z aposteriorního rozdělení:
<1.43769, 1.93272>
```

Porovnáním intervalů zjistíme, že apriorní rozdělení poskytuje mnohem širší rozsah možných hodnot, zatímco aposteriorní rozdělení vykazuje výrazně užší interval se stejnou spolehlivostí. Obecně platí, že apriorní rozložení vychází z expertní informace, aniž by byla k dispozici konkrétní data. Naopak aposteriorní rozložení kombinuje expertní informaci s reálnými naměřenými daty. Výsledky tak dávají smysl: před započítáním skutečných hodnot je interval širší (s více možnými hodnotami parametru), a až po zahrnutí naměřených hodnot získáváme užší a přesnější interval.

2.1.4 4) Výběr dvou aposteriorních bodových odhadů parametru

V rámci bodových odhadů je možné vybrat průměr, medián nebo modus. Pokud jsou data nevychýlená, očekáváme, že tyto tři hodnoty budou ekvivalentní. V obecném smyslu platí, že průměr je citlivý na extrémní hodnoty, ale zahrnuje všechna data, což je užitečné v případě, kdy důvěřujeme rozumnému modelování a nemáme konkrétní důvody některým hodnotám nevěřit.

Medián reprezentuje střední hodnotu intervalu a je odolnější vůči extrémním hodnotám než průměr. Ukazuje hodnotu, pod a nad kterou leží stejné procento dat (tedy 50. percentil). V případě výrazných odchylek od normálního rozdělení může být medián lepším ukazatelem střední hodnoty než průměr. Nicméně v našem konkrétním případě pozorujeme velmi podobné hodnoty obou bodových odhadů, což naznačuje, že data pravděpodobně nejsou vychýlena.

[5]:
```
# prvni bodovy odhad parametru lambda
mean = alphaPost/betaPost

# druhy bodovy odhad parametru lambda
median = st.gamma.median(a=alphaPost, scale=1/betaPost)

print(f"Bodový odhad parametru lambda za použití střední hodnoty: {mean:.5f}")
print(f"Bodový odhad parametru lambda za použití mediánu: {median:.5f}")
```

```
Bodový odhad parametru lambda za použití střední hodnoty: 1.67619
Bodový odhad parametru lambda za použití mediánu: 1.67302
```

2.1.5 5) Výběr apriorního a aposteriorního bodového odhadu počtu pozorování

[6]:
```
observationsPrior = alphaPrior * (1 - betaPrior / (betaPrior + 1)) / (betaPrior␣
↪/ (betaPrior + 1))
observationsPost = alphaPost * (1 - betaPost / (betaPost + 1)) / (betaPost /␣
↪(betaPost + 1))
```

```
print(f"Apriorní očekávaný počet pozorování: {observationsPrior:.5f}")
print(f"Aposteriorní očekávaný počet pozorování: {observationsPost:.5f}")
```

```
Apriorní očekávaný počet pozorování: 2.00000
Aposteriorní očekávaný počet pozorování: 1.67619
```

Bodový odhad parametru lambda pomocí průměru vyšel na hodnotě 2, což odpovídá očekávanému výsledku, když předpokládáme 10 připojení za 5 ms. Nicméně, tento odhad nezohledňuje aktuální data, která vedou k odlišnému výsledku 1,67. Tato odchylka naznačuje, že skutečný průměrný počet připojení za 1 ms je pravděpodobně nižší než 2, jak bylo očekáváno.

2.2 b) Aproximace diskrétním rozdělením

2.2.1 1) Graf apriorní, aposteriorní hustotou a funkce věrohodnosti

Namísto zadané apriorní pravděpodobnosti máme nyní naměřená data. Pro získání hustoty pravděpodobnosti $h(b)$ nyní spočteme pro každou skupinu nejvyšší hodnotu.

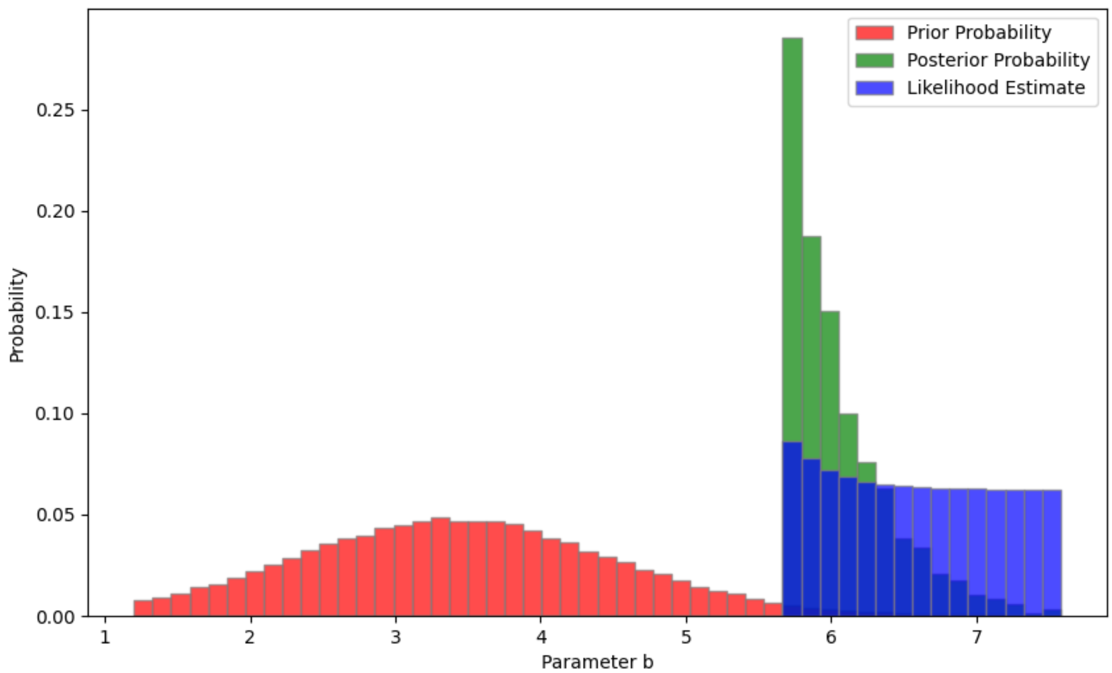
[7]:
```
observations = dataSet1["uloha_1 b)_pozorování"].dropna().values

# nalezneme maxima pro každou skupinu
data = dataSet1[['skupina', 'uloha_1 b)_prior']]  # Selecting columns for group␣
 ↪and task_1b_observation
groupedData = data.groupby("skupina")["uloha_1 b)_prior"]
maxValues = []

for k, v in groupedData:
    maxValues.append(max(v))

maxValues_inGroup = pd.DataFrame({"skupina": groupedData.groups.keys(),␣
 ↪"max_value": maxValues})
```

Tímto jsme dokázali získat pro každou skupinu největší hodnoty. Nyní je rozřadíme do binů, díky čemuž získáme apriorní rozdělení. Dále pracujeme s informacemi ze zadání, které říkají, že délka zpracování procesu v milisekundách má odseknuté normální rozdělení s určenými parametry, což nám říká, že rozdělení je normální pouze v určitém rozsahu. Pro výpočet věrohodnostní funkce využijme tedy odseknuté normální rozdělení. Pro každou z 50 možností parametru b získáváme pravděpodobnost (normalizovanou pomocí funkce SoftMax na intervalu $\langle 0, 1 \rangle$) v logaritmické doméně, což nám poskytuje informaci o pravděpodobnosti správnosti hodnoty parametru b.

V této fázi máme apriorní hustotu a funkci věrohodnosti. Bayesovým vzorcem poté vypočítáváme aposteriorní hustotu.

[8]:
```
# Creating prior histogram
numOfValuesInBin, binsEdges = np.histogram(data["uloha_1 b)_prior"], bins=50)
priorProb = numOfValuesInBin / numOfValuesInBin.sum()
binsCenters = binsEdges[:-1] + np.diff(binsEdges) / 2
```

```python
# Creating output
likelihoodEstimate = []

# Calculating likelihood for each bin center
mu = 3
sigma = 1
for binCenter in binsCenters:
    likelihoodEstimate.append(np.sum(truncnorm(a=(1 - mu) / sigma, b=(binCenter␣
 ↪- mu) / sigma, loc=mu, scale=sigma).logpdf(observations)))

# Softmax for converting likelihood to probabilities
likelihoodProbs = softmax(likelihoodEstimate)

# Creating posterior probabilities
posteriorEstimate = priorProb * likelihoodProbs
posteriorProbs = posteriorEstimate / np.sum(posteriorEstimate)

# Plotting results
plt.figure(figsize=(10, 6))
plt.hist(binsEdges[:-1], binsEdges, weights=priorProb, label='Prior␣
 ↪Probability', alpha=0.7, color='red', edgecolor='grey')
plt.hist(binsEdges[:-1], binsEdges, weights=posteriorProbs, label='Posterior␣
 ↪Probability', alpha=0.7, color='green', edgecolor='grey')
plt.hist(binsEdges[:-1], binsEdges, weights=likelihoodProbs, label='Likelihood␣
 ↪Estimate', alpha=0.7, color='blue', edgecolor='grey')
plt.xlabel('Parameter b')
plt.ylabel('Probability')
plt.legend()
plt.show()
```



2.2.2 2) 95% interval spolehlivosti parametru b

Nejprve seřaďme hodnoty parametru vzestupně podle pravděpodobnosti. Dále spočteme komulativní aposteriorní pravděpodobnosti. Tím dostaneme aposteriorní distribuční funkci. Z té jsme pak schopni najít hodnoty pro interval 95% spolehlivosti.

[9]:
```
# Kombinujte středy binů s odpovídajícími aposteriorními pravděpodobnostmi␣
↪pomocí funkce zip
# Seřaďte objekty sestupně podle druhé hodnoty (pravděpodobnosti) od největší␣
↪po nejmenší
sortedPost = sorted(zip(binsCenters, posteriorProbs), key=lambda x: x[1],␣
↪reverse=True)

# Spočítejte kumulativní (seřazené) aposteriorní pravděpodobnosti (kumulativní␣
↪součet pole),
# což nám dává kumulativní distribuční funkci aposteriorní pravděpodobnosti
cumulativeProbs = np.cumsum([prob for _, prob in sortedPost])

# Pomocí kumulativní distribuční funkce najděte rozsah 95% intervalu␣
↪spolehlivosti
# (argmax vrací index prvního prvku, který je větší nebo roven hledané hodnotě)
lowerBoundIndex = np.argmax(cumulativeProbs >= 0.025)
upperBoundIndex = np.argmax(cumulativeProbs >= 0.975)

# Indexy slouží k přístupu k hodnotám parametru b
```

```
lowerBound = sortedPost[lowerBoundIndex][0]
upperBound = sortedPost[upperBoundIndex][0]

print(f"95% confidence interval for parameter b from the posterior density:␣
 ↪<{lowerBound:.5f}, {upperBound:.5f}>")
```

```
95% confidence interval for parameter b from the posterior density: <5.72921,
7.00481>
```

2.2.3 3) Bodové odhady parametru b

Jako bodové odhady zvolíme průměr a modus. Pro výpočet průměru opět využijeme diskrétních hodnot. Modus spočítáme jako *argmax* z aposteriorní pravděpodobnostní funkce (diskretizované hustoty).

[10]:
```
# Bodové odhady parametru b
mean = np.sum(binsCenters * posteriorProbs)      # Váhovaný průměr (každý␣
 ↪parametr b má přiřazenou pravděpodobnost)
mode = binsCenters[np.argmax(posteriorProbs)]    # Parametr b s největší␣
 ↪pravděpodobností

print(f"Bodový odhad parametru b za použití průměru: {mean:.5f}")
print(f"Bodový odhad parametru b za použití modu: {mode:.5f}")
```

```
Bodový odhad parametru b za použití průměru: 6.04911
Bodový odhad parametru b za použití modu: 5.72921
```

3 2. úloha – Regrese

Nejprve se podívejme, jak vypadají data pro druhý dataset.

[11]:
```
print(dataSet2)
```

```
      OSType  ActiveUsers  InteractingPct  ScrollingPct  Ping [ms]
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
..       ...          ...             ...           ...        ...
497      iOS         5315          0.1974        0.8026         28
498    MacOS         1392          0.2373        0.7627         24
499      iOS         6014          0.8112        0.1888         54
500  Android         5118          0.2345        0.7655         39
501    MacOS         2660          0.9390        0.0610         55

[502 rows x 5 columns]
```

Nejprve budeme chtít dataset vyčistit od případných korelovaných dat. Pak bude třeba nalézt vychýlená data, která mohou nějak ovlivňovat model. Až poté budeme moci začít model vytvářet.

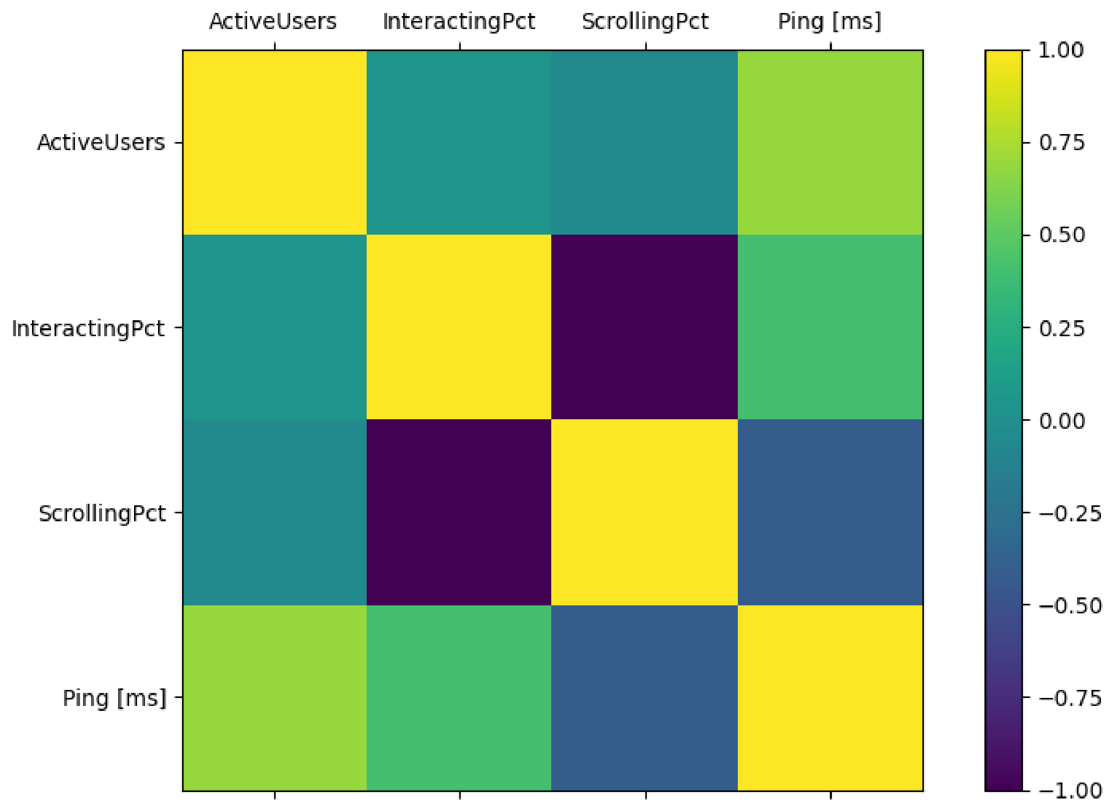
3.0.1 Čištění a úprava vstupních dat, příprava

[12]:
```
# Kontrola, zda `1 - InteractingPct = ScrollingPct`
check = 1 - dataSet2['InteractingPct'] - dataSet2['ScrollingPct']

# Výpis datového rámce s přidaným sloupcem 'Check'
print(check)
```

```
0      9.714451e-16
1      8.881784e-16
2      0.000000e+00
3      0.000000e+00
4      5.551115e-17
           …
497    0.000000e+00
498    8.881784e-16
499    9.714451e-16
500    1.110223e-16
501    5.551115e-17
Length: 502, dtype: float64
```

Jak z výpisu můžeme vidět, atributy `InteractingPct` a `ScrollingPct` jsou na sobě závislé (hodnoty jsou po provedení výpočtu velice nízké - a tedy jsou na sobě závislé), proto můžeme jeden atribut bez starostí odstranit. Než tak učiníme, ještě se na ně podíváme v grafu korelační matice, která nám může ukázat jednotlivé závislosti vizuálně.

[13]:
```
atributes = dataSet2.select_dtypes(include=[np.number])
plt.figure(figsize=(10, 6))
plt.matshow(atributes.corr(), fignum=1)
plt.xticks(range(len(atributes.columns)), atributes.columns)
plt.yticks(range(len(atributes.columns)), atributes.columns)
plt.colorbar()
plt.show()
```



3.1 1) Určení vhodného regresního modelu

3.1.1 Plný kvadratický model

Má následující tvar:

$ y= _1 + _2 OSType + _3 ActiveUsers + *4 InteractingPct +* 5 ScrollingPct + *6 OSType ActiveUsers +* 7 OSType InteractingPct + *8 OSType ScrollingPct +* 9 ActiveUsers InteractingPct + *{10} ActiveUsers ScrollingPct +* {11} InteractingPct ScrollingPct + *{12} OSType^2 +* {13} ActiveUsers^2 + *{14} InteractingPct^2 +* {15} ScrollingPct^2 $

V tomto modelu máme jednu kategoriální proměnnou, kterou budeme muset zakódovat, díky čemuž bude mít číselnou reprezentaci. Jedná se o *OSType*. Pro to využijeme *one-hot encoding*. V této proměnné se vyskytují čtyři hodnoty, ze kterých se stanou sloupce a vždy tak bude na řádku v tabulce právě jedna 1 u daného operačního systému. Tím nám tedy zanikne jeden sloupec - OSType, ale přibudou další čtyři.

Jak jsme si řekli výše, dva sloupečky jsou na sobě závislé a proto můžeme jeden ostranit. V konečném důstledku se nám tabulka rozšíří o 2 sloupce.

[14]:
```
# odstraneni ScrollingPct sloupce + prejmenovani Ping sloupce pro jednodussi␣
 ↪zachazeni
dataSet2Changed = dataSet2.drop('InteractingPct', axis=1)
```

```
dataSet2Changed = dataSet2Changed.rename(columns={'Ping [ms]': 'Ping'})

# one hot encoding
dataSet2_OneHot = pd.get_dummies(dataSet2Changed["OSType"]).
 ↪join(dataSet2Changed.drop("OSType", axis=1))

dataSet2_OneHot
```

[14]:

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

[502 rows x 7 columns]
```

3.1.2 Ořezávání atrubutů

Model ořežeme o atributy, druhé mocniny a kombinace jak jsme si řekli výše a podíváme se na výstup.

[15]:

```
fullModel = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + Android:ScrollingPct + \
                        MacOS:ActiveUsers + MacOS:ScrollingPct + \
                        Windows:ActiveUsers + Windows:ScrollingPct + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers) +␣
 ↪I(ScrollingPct*ScrollingPct)", data=dataSet2_OneHot)
results = fullModel.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.839
Method:                 Least Squares   F-statistic:                     187.9
Date:                Sun, 17 Dec 2023   Prob (F-statistic):          5.18e-186
Time:                        21:47:48   Log-Likelihood:                -1598.4
No. Observations:                 502   AIC:                             3227.
```

```
Df Residuals:                     487   BIC:                             3290.
Df Model:                          14
Covariance Type:            nonrobust
================================================================================
==================
                                   coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
Intercept                       33.7611      2.219     15.214      0.000
29.401      38.121
Android[T.True]                 -0.2195      2.405     -0.091      0.927
-4.945       4.506
MacOS[T.True]                    1.4257      2.135      0.668      0.505
-2.769       5.620
Windows[T.True]                  8.0239      2.277      3.524      0.000
3.550      12.498
ActiveUsers                      0.0059      0.001     10.491      0.000
0.005       0.007
Android[T.True]:ActiveUsers      0.0011      0.000      3.369      0.001
0.000       0.002
MacOS[T.True]:ActiveUsers        0.0025      0.000      8.370      0.000
0.002       0.003
Windows[T.True]:ActiveUsers      0.0003      0.000      1.021      0.308
-0.000       0.001
ScrollingPct                   -30.4225      4.404     -6.908      0.000
-39.076     -21.769
Android[T.True]:ScrollingPct     0.2678      2.691      0.100      0.921
-5.020       5.556
MacOS[T.True]:ScrollingPct       0.6244      2.440      0.256      0.798
-4.171       5.420
Windows[T.True]:ScrollingPct    -0.1582      2.634     -0.060      0.952
-5.333       5.016
ActiveUsers:ScrollingPct         0.0031      0.000      8.532      0.000
0.002       0.004
I(ActiveUsers * ActiveUsers)  -4.17e-07    4.4e-08     -9.469      0.000
-5.03e-07    -3.3e-07
I(ScrollingPct * ScrollingPct)  -3.7258      3.492     -1.067      0.287
-10.587       3.135
==============================================================================
Omnibus:                      228.442   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3152.488
Skew:                           1.603   Prob(JB):                         0.00
Kurtosis:                      14.851   Cond. No.                     1.01e+09
==============================================================================

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

```
specified.
[2] The condition number is large, 1.01e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Lineárně závislé atributy jsme odstranili už ze začátku, avšak jsme se nevyhnuli multikolinearitě, což nám napovídá i OLS model. Jedná se samozřejmě o jednotlivé sloupce vytvořené z OSType atributu. Dále pak jakýkoli jednoduchý atribut na druhou, protože to je na sobě jednoznačně závislé.

Kdyby byl model co nejmenší, determinant matice plánu by měl být nulový a jednotlivé koeficienty by z ostatních nemělo být možno odhadnout.

Pro zjištění multikolinearity spočítáme VIF. Díky výsledku z tohoto faktoru můžeme odhadnout, jak moc jsou prediktory závislé. Pokud je hodnota faktoru do 10, prediktory jsou nezávislé. Pokud je to nad 10, pak už závislé jsou a musíme s tím něco dělat, protože to nám může ovlivňovat výsledky.

[16]:
```python
# Extrahování exogenních proměnných z plného modelu
exogVariables = pd.DataFrame(fullModel.exog, columns=fullModel.exog_names)

# Výpočet Variance Inflation Factor (VIF) pro každou exogenní proměnnou
vifValues = pd.Series([variance_inflation_factor(exogVariables.values, i)
                       for i in range(exogVariables.shape[1])],
                      index=exogVariables.columns)

# Vytvoření DataFrame pro hodnoty VIF
vifDataFrame = vifValues.to_frame()
vifDataFrame.columns = ['VIF']

# Zobrazení DataFrame s hodnotami VIF
print(vifDataFrame)
```

```
                                     VIF
Intercept                      70.251414
Android[T.True]                14.305992
MacOS[T.True]                  12.960231
Windows[T.True]                14.473358
ActiveUsers                    29.055991
Android[T.True]:ActiveUsers    10.021529
MacOS[T.True]:ActiveUsers       9.485078
Windows[T.True]:ActiveUsers     9.286836
ScrollingPct                   24.195197
Android[T.True]:ScrollingPct    6.730996
MacOS[T.True]:ScrollingPct      6.039982
Windows[T.True]:ScrollingPct    7.188844
ActiveUsers:ScrollingPct        8.705628
I(ActiveUsers * ActiveUsers)   22.499134
I(ScrollingPct * ScrollingPct) 16.422293
```

Po vzoru demonstračního cvičení odstraníme multikolinearitu standardizací do rozsahu $\langle -1, 1 \rangle$.

[17]:
```
dataSet2_OneHot = dataSet2_OneHot.astype(float)
mins = dataSet2_OneHot.min(axis=0)
maxes = dataSet2_OneHot.max(axis=0)
scaledAndCenteredData  = (dataSet2_OneHot - mins) / (maxes - mins) * 2 - 1

scaledAndCenteredData
```

[17]:
```
     Android  MacOS  Windows  iOS  ActiveUsers  ScrollingPct      Ping
0       -1.0   -1.0     -1.0  1.0    -0.191837     -0.658752 -0.088608
1       -1.0   -1.0     -1.0  1.0     0.509388      0.307484 -0.113924
2       -1.0   -1.0      1.0 -1.0     0.775918      0.564573  0.113924
3        1.0   -1.0     -1.0 -1.0     0.778980      0.841900  0.139241
4       -1.0    1.0     -1.0 -1.0     0.919592     -0.458171  0.645570
..       ...    ...      ...  ...          ...           ...       ...
497     -1.0   -1.0     -1.0  1.0     0.053469      0.605450 -0.569620
498     -1.0    1.0     -1.0 -1.0    -0.747143      0.525498 -0.670886
499     -1.0   -1.0     -1.0  1.0     0.196122     -0.624487  0.088608
500      1.0   -1.0     -1.0 -1.0     0.013265      0.531109 -0.291139
501     -1.0    1.0     -1.0 -1.0    -0.488367     -0.880573  0.113924

[502 rows x 7 columns]
```

Nyní se podívejme na OLS model znovu.

[18]:
```
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + Android:ScrollingPct + \
                        MacOS:ActiveUsers + MacOS:ScrollingPct + \
                        Windows:ActiveUsers + Windows:ScrollingPct + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers) +␣
 ↪I(ScrollingPct*ScrollingPct)", data=scaledAndCenteredData )
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.839
Method:                 Least Squares   F-statistic:                     187.9
Date:                Sun, 17 Dec 2023   Prob (F-statistic):          5.18e-186
Time:                        21:47:48   Log-Likelihood:                 247.09
No. Observations:                 502   AIC:                            -464.2
Df Residuals:                     487   BIC:                            -400.9
Df Model:                          14
Covariance Type:            nonrobust
================================================================================
==================
```

```
                                    coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
Intercept                         0.2126      0.017     12.499      0.000
0.179       0.246
Android                           0.0666      0.010      6.515      0.000
0.047       0.087
MacOS                             0.1791      0.010     18.701      0.000
0.160       0.198
Windows                           0.1195      0.010     12.379      0.000
0.101       0.139
ActiveUsers                       0.6346      0.027     23.612      0.000
0.582       0.687
ScrollingPct                     -0.2298      0.023     -9.860      0.000
-0.276      -0.184
Android:ActiveUsers               0.0656      0.019      3.369      0.001
0.027       0.104
Android:ScrollingPct              0.0017      0.017      0.100      0.921
-0.032       0.035
MacOS:ActiveUsers                 0.1523      0.018      8.370      0.000
0.117       0.188
MacOS:ScrollingPct                0.0039      0.015      0.256      0.798
-0.026       0.034
Windows:ActiveUsers               0.0184      0.018      1.021      0.308
-0.017       0.054
Windows:ScrollingPct             -0.0010      0.017     -0.060      0.952
-0.034       0.032
ActiveUsers:ScrollingPct          0.1911      0.022      8.532      0.000
0.147       0.235
I(ActiveUsers * ActiveUsers)     -0.2535      0.027     -9.469      0.000
-0.306      -0.201
I(ScrollingPct * ScrollingPct)   -0.0235      0.022     -1.067      0.287
-0.067       0.020
==============================================================================
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.                         7.89
==============================================================================

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

Hláška o velké multikolinearitě zmizela, takže standardizace opravdu pomohla.

V následujících krocích budeme postupně odstraňovat problematické prediktory (tedy ty s nejvyšší hodnotou ve sloupci `P>|t|`). Musíme však sledovat hodnotu `R-squared`, která musí být nejhůře stejně tak dobrá jako v předchozím kroku.

[19]:
```
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +
 ↪ScrollingPct + \
                        Android:ActiveUsers + Android:ScrollingPct + \
                        MacOS:ActiveUsers + MacOS:ScrollingPct + \
                        Windows:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers) +
 ↪I(ScrollingPct*ScrollingPct)", data=scaledAndCenteredData )
results = model.fit()
print(results.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, 17 Dec 2023   Prob (F-statistic):          3.57e-187
Time:                        21:47:48   Log-Likelihood:                 247.09
No. Observations:                 502   AIC:                            -466.2
Df Residuals:                     488   BIC:                            -407.1
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
==================
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
Intercept                       0.2125      0.017     12.550      0.000
0.179       0.246
Android                         0.0666      0.010      6.533      0.000
0.047       0.087
MacOS                           0.1790      0.010     18.767      0.000
0.160       0.198
Windows                         0.1195      0.010     12.472      0.000
0.101       0.138
ActiveUsers                     0.6346      0.027     23.654      0.000
0.582       0.687
ScrollingPct                   -0.2288      0.016    -14.168      0.000
-0.261      -0.197
Android:ActiveUsers             0.0656      0.019      3.372      0.001
0.027       0.104
Android:ScrollingPct            0.0022      0.015      0.146      0.884
-0.027       0.031
MacOS:ActiveUsers               0.1523      0.018      8.378      0.000
0.117       0.188
```

```
MacOS:ScrollingPct              0.0044      0.013      0.337      0.736
-0.021       0.030
Windows:ActiveUsers             0.0185      0.018      1.028      0.305
-0.017       0.054
ActiveUsers:ScrollingPct        0.1910      0.022      8.549      0.000
0.147       0.235
I(ActiveUsers * ActiveUsers)   -0.2534      0.027     -9.479      0.000
-0.306      -0.201
I(ScrollingPct * ScrollingPct) -0.0235      0.022     -1.067      0.286
-0.067       0.020
==============================================================================
Omnibus:                      228.409   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3151.111
Skew:                           1.603   Prob(JB):                         0.00
Kurtosis:                      14.848   Cond. No.                         7.89
==============================================================================

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

[20]:
```python
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + MacOS:ScrollingPct + \
                        Windows:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers) +␣
 ↪I(ScrollingPct*ScrollingPct)", data=scaledAndCenteredData )
results = model.fit()
print(results.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, 17 Dec 2023   Prob (F-statistic):          2.38e-188
Time:                        21:47:48   Log-Likelihood:                 247.07
No. Observations:                 502   AIC:                            -468.1
Df Residuals:                     489   BIC:                            -413.3
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
==================
                                     coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
```

```
------------------
Intercept                          0.2126      0.017     12.593      0.000
0.179       0.246
Android                            0.0667      0.010      6.574      0.000
0.047       0.087
MacOS                              0.1791      0.010     18.809      0.000
0.160       0.198
Windows                            0.1195      0.010     12.490      0.000
0.101       0.138
ActiveUsers                        0.6346      0.027     23.678      0.000
0.582       0.687
ScrollingPct                      -0.2303      0.012    -18.673      0.000
-0.255      -0.206
Android:ActiveUsers                0.0656      0.019      3.373      0.001
0.027       0.104
MacOS:ActiveUsers                  0.1523      0.018      8.389      0.000
0.117       0.188
MacOS:ScrollingPct                 0.0037      0.012      0.305      0.760
-0.020       0.028
Windows:ActiveUsers                0.0185      0.018      1.034      0.302
-0.017       0.054
ActiveUsers:ScrollingPct           0.1914      0.022      8.657      0.000
0.148       0.235
I(ActiveUsers * ActiveUsers)      -0.2534      0.027     -9.490      0.000
-0.306      -0.201
I(ScrollingPct * ScrollingPct)    -0.0236      0.022     -1.075      0.283
-0.067       0.020
==============================================================================
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.                         7.89
==============================================================================

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

[21]:
```
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + \
                        Windows:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers) +␣
 ↪I(ScrollingPct*ScrollingPct)", data=scaledAndCenteredData )
results = model.fit()
```

```python
print(results.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, 17 Dec 2023   Prob (F-statistic):         1.57e-189
Time:                        21:47:48   Log-Likelihood:                 247.03
No. Observations:                 502   AIC:                            -470.1
Df Residuals:                     490   BIC:                            -419.4
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
==================
                                     coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
Intercept                          0.2126      0.017     12.602      0.000
0.179       0.246
Android                            0.0667      0.010      6.586      0.000
0.047       0.087
MacOS                              0.1790      0.010     18.824      0.000
0.160       0.198
Windows                            0.1195      0.010     12.506      0.000
0.101       0.138
ActiveUsers                        0.6349      0.027     23.721      0.000
0.582       0.687
ScrollingPct                      -0.2317      0.011    -20.223      0.000
-0.254      -0.209
Android:ActiveUsers                0.0656      0.019      3.380      0.001
0.027       0.104
MacOS:ActiveUsers                  0.1524      0.018      8.398      0.000
0.117       0.188
Windows:ActiveUsers                0.0187      0.018      1.042      0.298
-0.017       0.054
ActiveUsers:ScrollingPct           0.1914      0.022      8.662      0.000
0.148       0.235
I(ActiveUsers * ActiveUsers)      -0.2536      0.027     -9.504      0.000
-0.306      -0.201
I(ScrollingPct * ScrollingPct)    -0.0238      0.022     -1.087      0.278
-0.067       0.019
==============================================================================
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.                         7.87
==============================================================================

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

[22]:

```
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers) +␣
↪I(ScrollingPct*ScrollingPct)", data=scaledAndCenteredData )
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     264.5
Date:                Sun, 17 Dec 2023   Prob (F-statistic):         1.62e-190
Time:                        21:47:48   Log-Likelihood:                 246.47
No. Observations:                 502   AIC:                            -470.9
Df Residuals:                     491   BIC:                            -424.5
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
==================
                                   coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
Intercept                        0.2130      0.017     12.632      0.000
0.180       0.246
Android                          0.0671      0.010      6.620      0.000
0.047       0.087
MacOS                            0.1793      0.010     18.860      0.000
0.161       0.198
Windows                          0.1206      0.010     12.691      0.000
0.102       0.139
ActiveUsers                      0.6158      0.020     31.484      0.000
0.577       0.654
ScrollingPct                    -0.2320      0.011    -20.255      0.000
-0.255      -0.210
Android:ActiveUsers              0.0556      0.017      3.299      0.001
0.022       0.089
```

```
MacOS:ActiveUsers                  0.1424      0.015      9.241      0.000
0.112       0.173
ActiveUsers:ScrollingPct           0.1902      0.022      8.620      0.000
0.147       0.234
I(ActiveUsers * ActiveUsers)      -0.2517      0.027     -9.454      0.000
-0.304      -0.199
I(ScrollingPct * ScrollingPct)    -0.0234      0.022     -1.069      0.285
-0.066       0.020
==============================================================================
Omnibus:                      227.432   Durbin-Watson:                   1.930
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3143.418
Skew:                           1.593   Prob(JB):                         0.00
Kurtosis:                      14.838   Cond. No.                         6.23
==============================================================================

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

[23]:

```python
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers)", data=scaledAndCenteredData␣
 ↪)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     293.7
Date:                Sun, 17 Dec 2023   Prob (F-statistic):         1.62e-191
Time:                        21:47:48   Log-Likelihood:                 245.89
No. Observations:                 502   AIC:                            -471.8
Df Residuals:                     492   BIC:                            -429.6
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
================
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
Intercept                       0.2050      0.015     13.557      0.000
0.175       0.235
```

```
Android                              0.0675      0.010      6.673      0.000
0.048       0.087
MacOS                                0.1789      0.010     18.829      0.000
0.160       0.198
Windows                              0.1214      0.009     12.804      0.000
0.103       0.140
ActiveUsers                          0.6155      0.020     31.467      0.000
0.577       0.654
ScrollingPct                        -0.2323      0.011    -20.285      0.000
-0.255      -0.210
Android:ActiveUsers                  0.0542      0.017      3.225      0.001
0.021       0.087
MacOS:ActiveUsers                    0.1423      0.015      9.232      0.000
0.112       0.173
ActiveUsers:ScrollingPct             0.1903      0.022      8.621      0.000
0.147       0.234
I(ActiveUsers * ActiveUsers)        -0.2511      0.027     -9.432      0.000
-0.303      -0.199
==============================================================================
Omnibus:                      228.381   Durbin-Watson:                   1.925
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3196.157
Skew:                           1.598   Prob(JB):                         0.00
Kurtosis:                      14.941   Cond. No.                         5.97
==============================================================================

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

Nyní již u žádného koeficientu nevidíme čísla výrazně vyšší než 0,0. Model jsme tímto zjednodušili, ale podíváme-li se na koeficient determinizace, nezměnili jsme schopnost model predikovat.

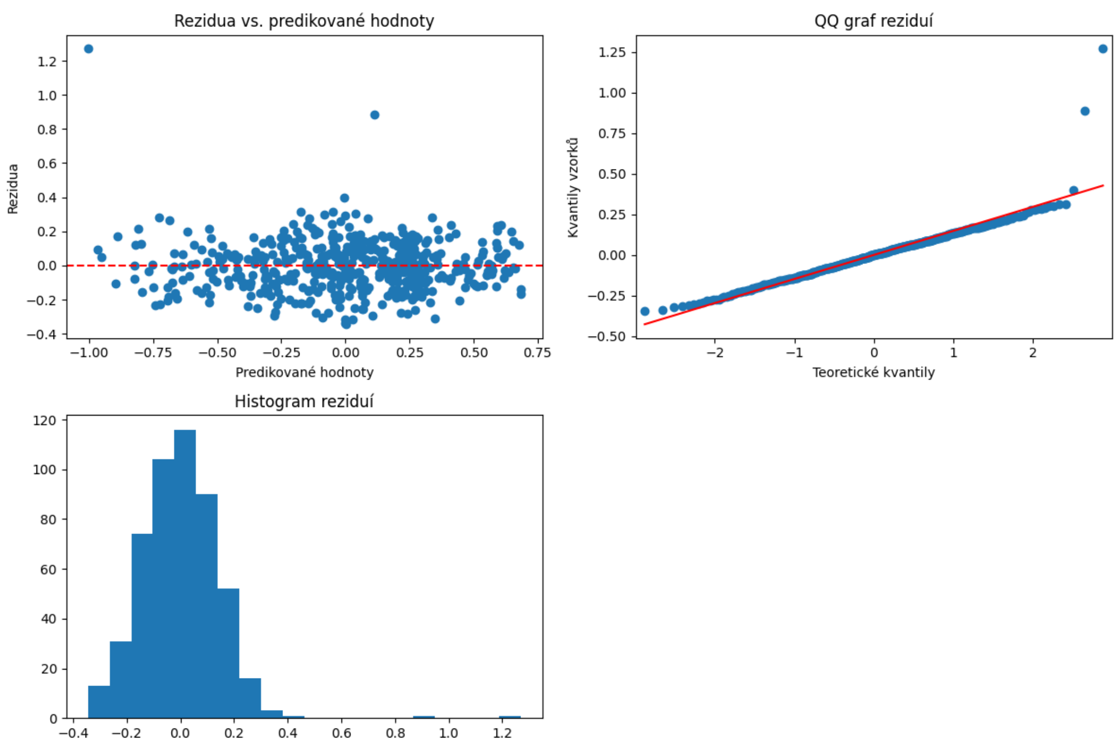
Statistika `Durbin-Watson` naznačuje, že za sebou jdoucí rezidua mají minimální pozitivní vzájemnou korelaci, což znamená, že nejsou významně autokorelovaná a navzájem se příliš neovlivňují. Ideální hodnota této statistiky je 2, a my se k ní přibližujeme.

Na druhé straně, statistiky `Omnibus` a `Jarque-Bera` vykazují vysoké hodnoty, což nám ukazuje, že rezidua nemají normální rozdělení. Toto může být způsobeno existencí několika vlivných bodů, se kterými jsme se dosud nezabývali. Pokud identifikujeme nějaký vlivný bod a považujeme ho za problematický, můžeme ho odstranit, což by mohlo zlepšit kvalitu našeho regresního modelu.

3.1.3 Určení vhodnosti dat pro regresní model

[24]:
```python
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# Rezidua vs. predikovane hodnoty
axs[0, 0].scatter(results.fittedvalues, results.resid)
axs[0, 0].axhline(y=0, color="red", linestyle="--")
axs[0, 0].set_title('Rezidua vs. predikované hodnoty')
```

```
axs[0, 0].set_xlabel('Predikované hodnoty')
axs[0, 0].set_ylabel('Rezidua')

# QQ graf rezidui
splt.qqplot(results.resid, line='s', ax=axs[0, 1])
axs[0, 1].set_title('QQ graf reziduí')
axs[0, 1].set_xlabel('Teoretické kvantily')
axs[0, 1].set_ylabel('Kvantily vzorků')

# histogram rezidui
axs[1, 0].hist(results.resid, bins=20)
axs[1, 0].set_title('Histogram reziduí')

# odstraneni prazdneho subplotu
fig.delaxes(axs[1, 1])

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



Z grafů můžeme vypozorovat dva vlivné body, které narušují jak normální rozdělení, tak celkovou kvalitu modelu. Pokud se jedná o chybná data, pak je odstraníme a aktualizujeme náš ořezaný/zjednodušený model.

```
[25]: # Identifikace odlehlych hodnot
outliers_indices = results.resid[results.resid > 0.5].index

for index in outliers_indices:
    print(index)
    print(str(dataSet2.iloc[index]) + "\n")
```

```
255
OSType            Windows
ActiveUsers          5513
InteractingPct     0.4912
ScrollingPct       0.5088
Ping [ms]              90
Name: 255, dtype: object

476
OSType             MacOS
ActiveUsers          153
InteractingPct    0.2111
ScrollingPct      0.7889
Ping [ms]             61
Name: 476, dtype: object
```

Tyto hodnoty můžeme odstranit, protože s nimi nechceme modelovat.

```
[26]: # odstraneni odlehlych hodnot (normalizovanych dat)
scaledAndCenteredData  = scaledAndCenteredData.drop(outliers_indices)
scaledAndCenteredData  = scaledAndCenteredData.reset_index(drop=True)

# odstraneni odlehlych hodnot (puvodni data)
dataSet2 = dataSet2.drop(outliers_indices)
dataSet2 = dataSet2.reset_index(drop=True)
```

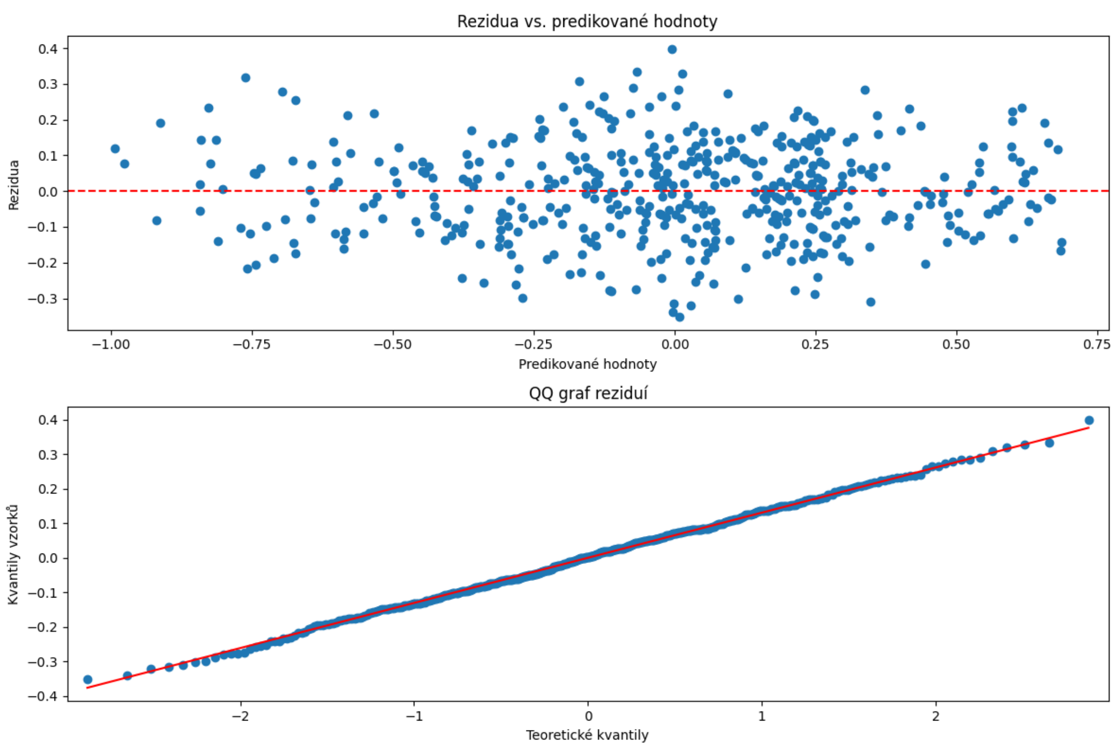
Nyní se znovu podívejme na model, jestli zůstal stále se stejnými, či lepšími hodnotami.

```
[27]: model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers)", data=scaledAndCenteredData␣
 ↪)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                   Ping   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.875
Method:                 Least Squares   F-statistic:                     388.1
Date:                Sun, 17 Dec 2023   Prob (F-statistic):          1.43e-216
Time:                        21:47:49   Log-Likelihood:                 308.63
No. Observations:                 500   AIC:                            -597.3
Df Residuals:                     490   BIC:                            -555.1
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
================
                                 coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
Intercept                      0.2021      0.013     15.149      0.000
0.176       0.228
Android                        0.0669      0.009      7.505      0.000
0.049       0.084
MacOS                          0.1713      0.008     20.398      0.000
0.155       0.188
Windows                        0.1182      0.008     14.140      0.000
0.102       0.135
ActiveUsers                    0.6415      0.017     36.779      0.000
0.607       0.676
ScrollingPct                  -0.2383      0.010    -23.592      0.000
-0.258      -0.218
Android:ActiveUsers            0.0558      0.015      3.777      0.000
0.027       0.085
MacOS:ActiveUsers              0.1639      0.014     11.929      0.000
0.137       0.191
ActiveUsers:ScrollingPct       0.2087      0.020     10.693      0.000
0.170       0.247
I(ActiveUsers * ActiveUsers)  -0.2793      0.024    -11.764      0.000
-0.326      -0.233
==============================================================================
Omnibus:                        0.799   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.671   Jarque-Bera (JB):                0.865
Skew:                           0.002   Prob(JB):                        0.649
Kurtosis:                       2.796   Cond. No.                         6.06
==============================================================================

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

[28]:
```
fig, axs = plt.subplots(2, 1, figsize=(12, 8))

# Rezidua vs. predikovane hodnoty
axs[0].scatter(results.fittedvalues, results.resid)
axs[0].axhline(y=0, color="red", linestyle="--")
axs[0].set_title('Rezidua vs. predikované hodnoty')
axs[0].set_xlabel('Predikované hodnoty')
axs[0].set_ylabel('Rezidua')

# QQ graf rezidui
splt.qqplot(results.resid, line='s', ax=axs[1])
axs[1].set_title('QQ graf reziduí')
axs[1].set_xlabel('Teoretické kvantily')
axs[1].set_ylabel('Kvantily vzorků')

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



Model bude ještě třeba znormalizovat, ale jinak vidíme, že model neobsahuje žádné nevýznamné prediktory či odlehlé hodnoty a tedy - náš model je finální. Po normalizaci už můžeme napsat jeho finální podobu.

```
[29]: # opetovna normalizace (aktualizovanych) dat
minValues = scaledAndCenteredData.min(axis=0)
maxValues = scaledAndCenteredData.max(axis=0)
scaledAndCenteredData  = -1+2*((scaledAndCenteredData -minValues)/(maxValues -␣
 ↪minValues))
scaledAndCenteredData.head()

model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers)", data=scaledAndCenteredData␣
 ↪)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.875
Method:                 Least Squares   F-statistic:                     388.1
Date:                Sun, 17 Dec 2023   Prob (F-statistic):          1.43e-216
Time:                        21:47:50   Log-Likelihood:                 269.14
No. Observations:                 500   AIC:                            -518.3
Df Residuals:                     490   BIC:                            -476.1
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
================
                                 coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
Intercept                      0.3600      0.014     25.018      0.000
0.332       0.388
Android                        0.0777      0.009      8.205      0.000
0.059       0.096
MacOS                          0.2011      0.009     22.300      0.000
0.183       0.219
Windows                        0.1279      0.009     14.140      0.000
0.110       0.146
ActiveUsers                    0.5840      0.017     35.312      0.000
0.551       0.616
ScrollingPct                  -0.2379      0.011    -21.964      0.000
-0.259      -0.217
Android:ActiveUsers            0.0551      0.015      3.777      0.000
```

```
0.026       0.084
MacOS:ActiveUsers               0.1616      0.014     11.929      0.000
0.135       0.188
ActiveUsers:ScrollingPct        0.2059      0.019     10.693      0.000
0.168       0.244
I(ActiveUsers * ActiveUsers)   -0.2511      0.021    -11.764      0.000
-0.293      -0.209
==============================================================================
Omnibus:                        0.799   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.671   Jarque-Bera (JB):                0.865
Skew:                           0.002   Prob(JB):                        0.649
Kurtosis:                       2.796   Cond. No.                         5.03
==============================================================================

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

Výsledná **rovnice modelu** pro **normalizované vstupy** tedy je:

$ Ping = 0.36 + 0.0777 Android + 0.2011 MacOS + 0.1279 Windows + 0.584 ActiveUsers - 0.2379 ScrollingPct + 0.0551 Android ActiveUsers + 0.1616 MacOS ActiveUsers + 0.2059 ActiveUsers ScrollingPct - 0.2511 ActiveUsers ActiveUsers $

3.2 2) Identifikace parametrů s nejproblematičtější hodnotou odezvy

Jako identifikaci problematických hodnot rozumím najití najití takové kombinace, u které model vygeneruje buďto minimální, nebo maximální výstup. V našem případě tedy hodnotu pingu.

[30]:
```
# Predicting ping values using the regression results
predictedPings = results.predict(scaledAndCenteredData )

# Finding the row index for the maximum and minimum predicted ping values
maxPingIndex = predictedPings.idxmax()
minPingIndex = predictedPings.idxmin()

# Extracting the corresponding input values for the maximum and minimum ping
maxPingInputs = dataSet2.loc[maxPingIndex]
minPingInputs = dataSet2.loc[minPingIndex]

# Displaying the results
print("Kombinace pro maximální Ping:")
print(maxPingInputs)

print("\n Kombinace pro minimální Ping:")
print(minPingInputs)
```

```
Kombinace pro maximální Ping:
OSType            MacOS
```

```
ActiveUsers        9657
InteractingPct    0.973
ScrollingPct      0.027
Ping [ms]            72
Name: 10, dtype: object

 Kombinace pro minimální Ping:
OSType              iOS
ActiveUsers        1128
InteractingPct    0.103
ScrollingPct      0.897
Ping [ms]            16
Name: 315, dtype: object
```

3.3 3) Odhad hodnoty odezvy uživatele s Windows

[31]:

```
# Výpočet průměrných hodnot standardizovaného DataFrame
dfMeanStandardized = scaledAndCenteredData.mean(axis=0).to_frame().T

# Definování výchozích hodnot pro typy OS
defaultValues = {"Windows": 1, "Android": -1, "MacOS": -1, "iOS": -1}

# Nastavení výchozích hodnot pro typy OS
dfMeanStandardized = dfMeanStandardized.assign(**defaultValues)

# Provedení predikce pomocí objektu výsledků
prediction = results.get_prediction(dfMeanStandardized)

# Extrahování rámce shrnutí pro hladinu důvěry 95 %
summaryDF = prediction.summary_frame(alpha=0.05)

# Přeškálování summaryDF na původní měřítko
scaledSummaryDF = (summaryDF + 1) / 2 * (maxes["Ping"] - mins["Ping"]) +␣
↪mins["Ping"]

# Zobrazení dolní a horní hranice intervalu spolehlivosti
ciLower = scaledSummaryDF['mean_ci_lower'][0]
ciUpper = scaledSummaryDF['mean_ci_upper'][0]
print(f"Konfidenční interval pingu při průměrném nastavení ostatních parametrů:␣
↪<{ciLower}, {ciUpper}>")

# Zobrazení dolní a horní hranice intervalu predikce
obsCILower = scaledSummaryDF['obs_ci_lower'][0]
obsCIUpper = scaledSummaryDF['obs_ci_upper'][0]
print(f"Predikční interval pingu při průměrném nastavení ostatních parametrů:␣
↪<{obsCILower}, {obsCIUpper}>")
```

```
Konfidenční interval pingu při průměrném nastavení ostatních parametrů:
<57.47756231590348, 59.708011708365085>
Predikční interval pingu při průměrném nastavení ostatních parametrů:
<47.46288256704841, 69.72269145722015>
```

3.4 4) Vhodnost modelu

[32]:
```python
model = smf.ols(formula="Ping ~ Android + MacOS + Windows + ActiveUsers +␣
 ↪ScrollingPct + \
                        Android:ActiveUsers + \
                        MacOS:ActiveUsers + \
                        ActiveUsers:ScrollingPct + \
                        I(ActiveUsers*ActiveUsers)", data=scaledAndCenteredData␣
 ↪)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Ping   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.875
Method:                 Least Squares   F-statistic:                     388.1
Date:                Sun, 17 Dec 2023   Prob (F-statistic):          1.43e-216
Time:                        21:47:50   Log-Likelihood:                 269.14
No. Observations:                 500   AIC:                            -518.3
Df Residuals:                     490   BIC:                            -476.1
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
================
                                   coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
Intercept                        0.3600      0.014     25.018      0.000
0.332       0.388
Android                          0.0777      0.009      8.205      0.000
0.059       0.096
MacOS                            0.2011      0.009     22.300      0.000
0.183       0.219
Windows                          0.1279      0.009     14.140      0.000
0.110       0.146
ActiveUsers                      0.5840      0.017     35.312      0.000
0.551       0.616
ScrollingPct                    -0.2379      0.011    -21.964      0.000
-0.259      -0.217
Android:ActiveUsers              0.0551      0.015      3.777      0.000
```

```
0.026       0.084
MacOS:ActiveUsers              0.1616      0.014     11.929      0.000
0.135       0.188
ActiveUsers:ScrollingPct       0.2059      0.019     10.693      0.000
0.168       0.244
I(ActiveUsers * ActiveUsers)  -0.2511      0.021    -11.764      0.000
-0.293      -0.209
==============================================================================
Omnibus:                        0.799   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.671   Jarque-Bera (JB):                0.865
Skew:                           0.002   Prob(JB):                        0.649
Kurtosis:                       2.796   Cond. No.                         5.03
==============================================================================

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

Pro zhodnocení modelu je nejlepší podívat se na kvalitu modelu, kterou dokážeme vyčíst z horní tabulky. S hodnotou R-squared se chceme přiblížit co nejvíce 1, a náš výsledek skoro 0,9 je obstojný. Veškeré koeficienty jsou pro nás klíčové, už žádný nemůžeme vyloučit. Nemáme žádné odlehlé hodnoty. Z tohoto hlediska je model dokončený a vhodný. Bylo by však záhodno model vyzkoušet na další testovací sadě, díky čemuž bychom mohli ověřit jeho korektnost.