xkadna00-msp-2-projekt

December 17, 2023

0.0.1 Načítanie knižníc

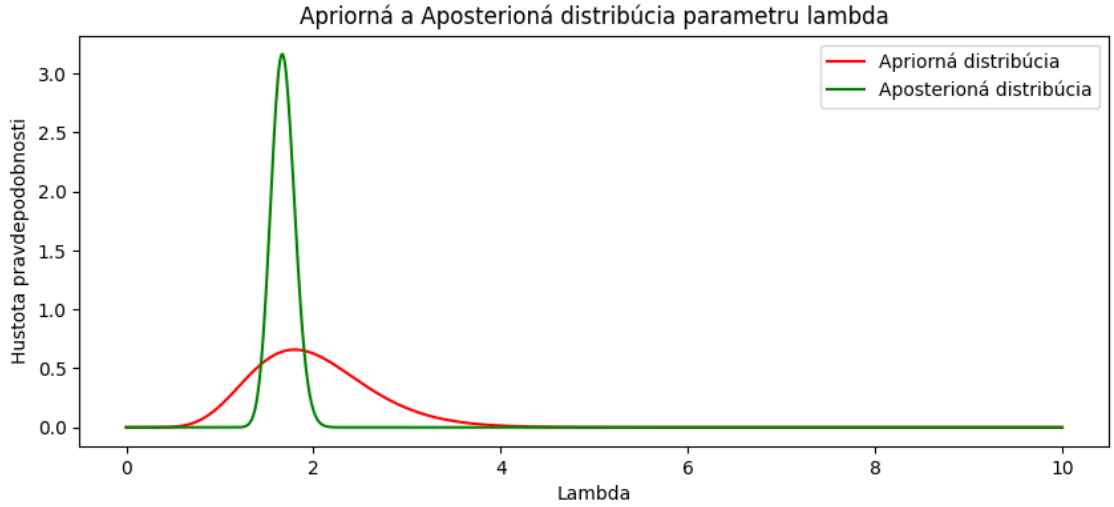
[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt
from tabulate import tabulate

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

0.0.2 Úloha 1 a)

[2]:
```
#načítanie dát pre úlohu 1
df1 = pd.read_excel('Projekt-2_Data.xlsx', sheet_name="Úloha 1", nrows=100)
interval_per = 0.95

#apriorné parametre pre gamma distribúciu
pri_beta = 5
pri_alpha = 10

#aposteriorné parametre pre gamma distribúciu
pos_beta = pri_beta + len(df1['uloha_1 a)'])
pos_alpha = pri_alpha + df1['uloha_1 a)'].sum()

pos_alpha_beta = pos_alpha/pos_beta

#výpočet intervalu spoľahlivosti pre apriornú a aposteriornú distribúciu
low_pri, high_pri = stats.gamma.interval(interval_per, pri_alpha, scale=1/
 ↪pri_beta)
low_pos, high_pos = stats.gamma.interval(interval_per, pos_alpha, scale=1/
 ↪pos_beta)
```

```
#apriorná a aposteriorná gamma distribúcia
prior_distribution = stats.gamma(pri_alpha, scale=1/pri_beta)
posterior_distribution = stats.gamma(pos_alpha, scale=1/pos_beta)

x_values = np.linspace(0, 10, 1000)
```

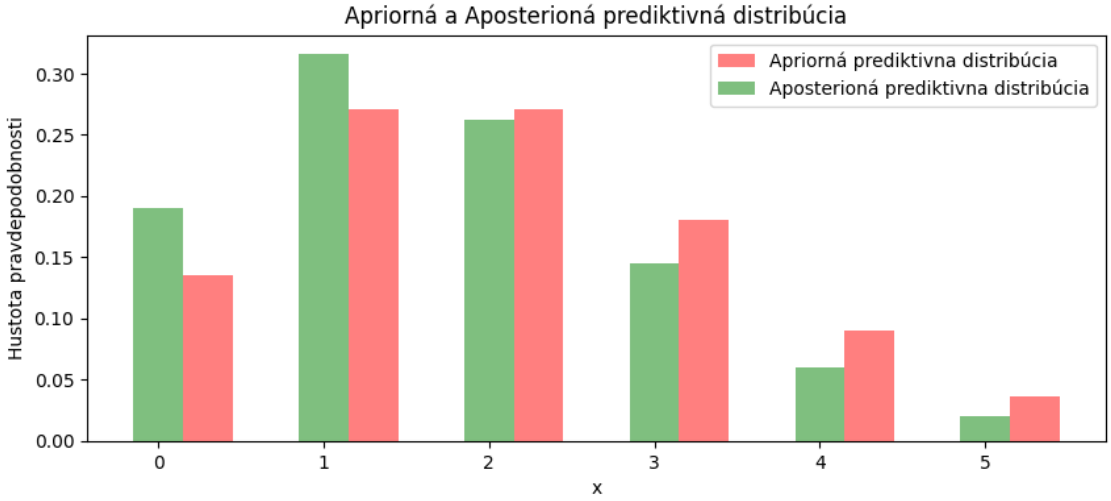
[3]:
```
#1a).1 a 1a).3
#apriorná a aposterioná hustota prametru Poissonovho rozdelenia
plt.figure(figsize=(10, 4))
plt.title('Apriorná a Aposterioná distribúcia parametru lambda')
plt.xlabel('Lambda')
plt.ylabel('Hustota pravdepodobnosti')

plt.plot(x_values, prior_distribution.pdf(x_values), label='Apriorná␣
 ↪distribúcia', color='red')
plt.plot(x_values, posterior_distribution.pdf(x_values), label='Aposterioná␣
 ↪distribúcia', color='green')

plt.legend()
plt.show()

print(f"apriorný interval: {low_pri} - {high_pri}" )
print(f"aposteriorný interval: {low_pos} {high_pos}")
```




```
apriorný interval: 0.9590777392264868 - 3.416960690283833
aposteriorný interval: 1.4376938284869922 1.9327207471868797
```

[4]:
```
#1a).2
```

```
#apriorná a aposterioná prediktivna hustota pozorovania x za jeden časový␣
 ↪interval
plt.figure(figsize=(10, 4))
plt.title('Apriorná a Aposterioná prediktivná distribúcia')
plt.xlabel('x')
plt.ylabel('Hustota pravdepodobnosti')

#vierohodnostná funkcia
x = [0,1,2,3,4,5]
likelihood_func = lambda x, lambda_param: stats.poisson.pmf(x, lambda_param)

#výpočet apriornej prediktívnej distribúcie pomocou pôvodnej lamby
aprior_dis = likelihood_func(x, pri_alpha/pri_beta)

#výpočet novej lambdy
new_lambda = df1['uloha_1 a)'].mean()

#výpočet aspoteriórnej prediktívnej distribúcie pomocou pôvodnej lamby
apos_dis = likelihood_func(x,new_lambda)

x = np.arange(0, max(df1['uloha_1 a)']) + 1)
plt.bar(x+0.3, aprior_dis,width = 0.3, alpha=0.5, label='Apriorná prediktivna␣
 ↪distribúcia', color='red')
plt.bar(x, apos_dis,width = 0.3, alpha=0.5, label='Aposterioná prediktivna␣
 ↪distribúcia', color='green')
plt.legend()
plt.show()
```



[5]:
```
#1a).4
#bodové odhady aposterioného parametra
pos_median = (pos_alpha - 1/3) / pos_beta

print(f"Aposteriony bodový odhad: \nStredná hodnota: {pos_alpha_beta} \nMedián:␣
 ↪{pos_median}")
```

```
Aposteriony bodový odhad:
Stredná hodnota: 1.6761904761904762
Medián: 1.6730158730158728
```

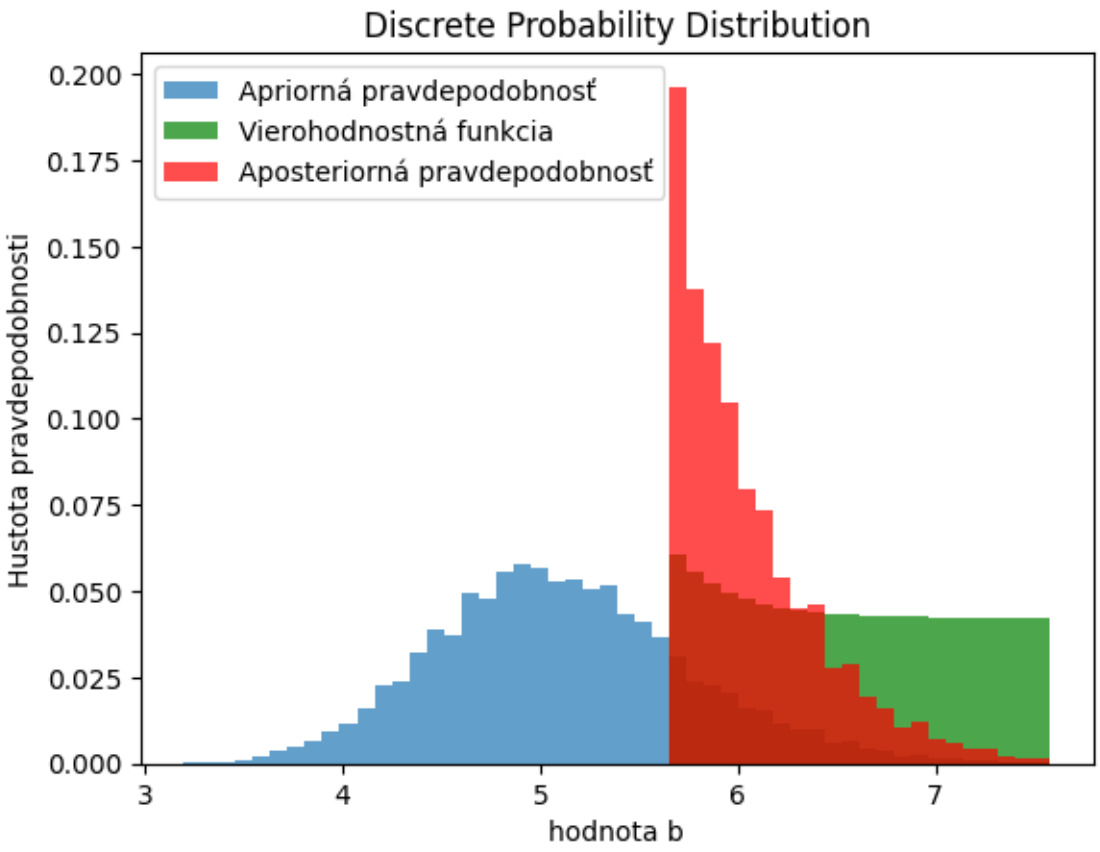
[6]:
```
#1a).5
#bodové odhady počtu pozorovaní
pri_mean = pri_alpha / pri_beta

print(f"Apriorný a aposterioný bodový odhad počtu pozorovania:")
print(f"Apriorná stredná hodnota: {pri_mean}")
print(f"Aposterioná stredná hodnota: {pos_alpha_beta}")
```

```
Apriorný a aposterioný bodový odhad počtu pozorovania:
Apriorná stredná hodnota: 2.0
Aposterioná stredná hodnota: 1.6761904761904762
```

0.0.3 Uloha 1 b)

[7]:
```
def cumulative_distribution_function(b, u=3, o=1):
    return stats.norm.cdf(b, loc=u, scale=o) - stats.norm.cdf(a_value, loc=u,␣
 ↪scale=o)

def standard_normal_distribution(x , u=3, o = 1):
    prob_density = (1/(np.sqrt(2*np.pi))) * np.exp(-0.5*((x-u)/o)**2)
    return prob_density

df = pd.read_excel('Projekt-2_Data.xlsx')
observations = df['uloha_1 b)_pozorování'].head(100)
a_value = 1
num_intervals = 50
num1 = 0
S = 0
viero = []


# Apriorná pravdepodobnosť
tmp_list = df.loc[df.groupby('skupina')['uloha_1 b)_prior'].idxmax()]
b = list(tmp_list["uloha_1 b)_prior"])

#tvorba histogramu
```

```
hist, bin_edges = np.histogram(b, bins=num_intervals, density=True)
midpoints = (bin_edges[1:] + bin_edges[:-1]) / 2

#nomralizácia
hist = hist / np.sum(hist)
plt.bar(midpoints, hist, width=(bin_edges[1] - bin_edges[0]), alpha=0.7,␣
 ↪label='Apriorná pravdepodobnosť')

# Vierohodnostná funkcia

#výpočet všetkých delitelov
for i in range(len(midpoints)):
  divider = cumulative_distribution_function(midpoints[i])
  num1 = hist[i]
  for j in observations:
    if j >= a_value and j <= midpoints[i]:
      num1 *= (standard_normal_distribution(j) / divider)
  S += num1

#výpočet nových b hdonôt ktoré sú väčšie ako hodnoty pozorovania
new_b = [i for i in midpoints if observations.max() <= i]

#výpočet preavdepodobnosti pre všekty pozorovania so zoznamom nových b hodnot
for i in range(len(new_b)):
  divider = cumulative_distribution_function(new_b[i])
  num1 = 1
  for j in observations:
      num1 *= (standard_normal_distribution(j) / divider)
  viero.append(num1)

#nomralizácia
normed_viero = viero / np.sum(viero)

plt.bar(new_b, normed_viero, width=(bin_edges[1] - bin_edges[0]), alpha=0.7,␣
 ↪label='Vierohodnostná funkcia', color='green')

# Aposteriorná pravdepodobnosť
aposterior = [normed_viero[i]*hist[28+i] for i in range(len(normed_viero))]
normed_apos = [i / sum(aposterior) for i in aposterior]
plt.bar(new_b, normed_apos, width=(bin_edges[1] - bin_edges[0]), alpha=0.7,␣
 ↪label='Aposteriorná pravdepodobnosť', color='red')

plt.xlabel('hodnota b')
plt.ylabel('Hustota pravdepodobnosti')
plt.title('Discrete Probability Distribution')
plt.legend()
plt.show()
```



[8]:
```
confidence_interval = np.percentile(new_b, [2.5, 97.5])
print(f"95% interval spolahlivosti: {confidence_interval[0]} -␣
  ↪{confidence_interval[1]}")
```

```
95% interval spolahlivosti: 5.739743979188164 - 7.488958117408135
```

[9]:
```
#Bodový odhad
print(f"Stredná hodnota: {np.mean(b)}")
print(f"Medián: {np.median(b)}")
```

```
Stredná hodnota: 5.117188290003788
Medián: 5.0716228091543805
```

0.0.4 Uloha 2

[10]:
```
def backward_elimination(formula, results, df):
  new_formula = formula
  new_results = results

  while len(new_results.pvalues) > 1 and new_results.pvalues.max() > 0.05:
```

```
    #nájednie parametra ktorý má najväčšiu p-hodnotu
    feature_to_remove = new_results.pvalues.idxmax()

    print(f'|p-value| {new_results.pvalues.max()}| |Removing |{new_results.
 ↪pvalues.idxmax()}| ')

    #odstránenie parametra z formule
    if feature_to_remove == "OSType_Android" or feature_to_remove ==␣
 ↪"OSType_iOS" or feature_to_remove == "OSType_Windows" or feature_to_remove␣
 ↪== "OSType_MacOS":
      for value in df.columns.tolist():
        new_formula = new_formula.replace(f'+{value}:{feature_to_remove} ', '')
        new_formula = new_formula.replace(f'+{feature_to_remove}:{value} ', '')
    new_formula = new_formula.replace(f'+{feature_to_remove} ', '')

    # znovuvytvorenie modelu a výsledkov pre novú formulu
    model = smf.ols(formula=new_formula, data=df)
    new_results = model.fit()

  return new_results, new_formula

def create_independent_df(df_encoded, correlation_threshold=0.7):
    correlation_matrix = df_encoded.corr()
    independent_variables = list(df_encoded.columns)

    for i in df_encoded.columns:
        # kontrola či premnná je stále v liste nezávislých hodnôt
        if i in independent_variables:
            #iterácie cez zostávajúce nezávislé premenné
            for j in independent_variables:
                # kontrola či premenné niesú rovnaké a či nepresahújú korelačnú␣
 ↪hranicu
                if i != j and abs(correlation_matrix.loc[i, j]) >␣
 ↪correlation_threshold:
                    independent_variables.remove(j)

    df_independent = df_encoded[independent_variables]
    df_variable_names = pd.DataFrame({"Variable Names": independent_variables})
    return df_independent, df_variable_names

def create_combination_multiplication(data):
  data2 = data
  result_matrix = pd.DataFrame()
  for col1 in data.columns:
      for col2 in data2.columns:
        if col1 != col2 and not ("OS" in col1 and "OS" in col2):
          result_matrix[f'{col1}:{col2}'] = data[col1] * data[col2]
```

```
      result_matrix[f'{col1}'] = data[col1]
      data2 = data2.drop(columns=[col1])

  for col1 in data.columns:
    result_matrix[f'{col1}'] = data[col1]

  return result_matrix
```

[11]:
```
df2 = pd.read_excel('Projekt-2_Data.xlsx', sheet_name='Úloha 2')

#one-hot enkódovanie (premena kategorických dát)
df_encoded = pd.get_dummies(df2, columns=['OSType'], prefix='OSType', dtype=␣
 ↪int, drop_first=True)

#normovanie nuemrických atribútov
data_to_normalize = df_encoded["ActiveUsers"].values.reshape(-1, 1)
scaler = StandardScaler()
normalized_data = scaler.fit_transform(data_to_normalize)
df_encoded_normalized = df_encoded
df_encoded_normalized["ActiveUsers"] = normalized_data

data_to_normalize = df_encoded_normalized["InteractingPct"].values.reshape(-1,␣
 ↪1)
scaler = StandardScaler()
normalized_data = scaler.fit_transform(data_to_normalize)
df_encoded_normalized["InteractingPct"] = normalized_data

#vytvorenie korelačnej matice z parametrov ktoré budú na pravej strane formule
tmp_df = df_encoded.drop("Ping [ms]",axis=1)
correlation_matrix = tmp_df.corr()

#odstránenie závislých dát
independent_vars1, df_independent = create_independent_df(correlation_matrix)

#vytvorenie kombinácií z nezaávislých premenných
combinations = create_combination_multiplication(independent_vars1)

#pridanie kvadratickej hodnoty ActiveUsers
combinations[f'I(ActiveUsers**2)'] = combinations["ActiveUsers"] ** 2

#vytvorenie formule z čisto nezaávislých dát
formula = 'Q("Ping [ms]") ~ '
for i in combinations.columns:
  formula += "+"+i+" "

#vytvorenie všetkých kombinácií dát pre model
all_combinations = create_combination_multiplication(df_encoded_normalized)
```

```
all_combinations[f'I(ActiveUsers**2)'] = df_encoded_normalized["ActiveUsers"]␣
 ↪** 2

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

#využitie spätnej eliminácie pre získanie finálneho tvaru fomule
results, formula = backward_elimination(formula, results, all_combinations)
model = smf.ols(formula=formula, data=all_combinations)
results = model.fit()
print("\nVýsledná rovnica: \n\nQ(\"Ping [ms]\") ~ +ActiveUsers:InteractingPct␣
 ↪+ActiveUsers:OSType_MacOS +ActiveUsers:OSType_Windows \
\n+ActiveUsers:OSType_iOS +ActiveUsers +InteractingPct +OSType_MacOS␣
 ↪+OSType_Windows +OSType_iOS +I(ActiveUsers**2)\n")
print(results.summary())
```

```
|p-value| 0.8864468379160608| |Removing |InteractingPct:OSType_MacOS|
|p-value| 0.7824377896785703| |Removing |InteractingPct:OSType_iOS|
|p-value| 0.8100814923990263| |Removing |InteractingPct:OSType_Windows|

Výsledná rovnica:

Q("Ping [ms]") ~ +ActiveUsers:InteractingPct +ActiveUsers:OSType_MacOS
+ActiveUsers:OSType_Windows
+ActiveUsers:OSType_iOS +ActiveUsers +InteractingPct +OSType_MacOS
+OSType_Windows +OSType_iOS +I(ActiveUsers**2)

                            OLS Regression Results
==============================================================================
Dep. Variable:         Q("Ping [ms]")   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     264.4
Date:                Sun, 17 Dec 2023   Prob (F-statistic):          1.69e-190
Time:                        20:11:17   Log-Likelihood:                -1599.1
No. Observations:                 502   AIC:                             3220.
Df Residuals:                     491   BIC:                             3267.
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
==============
                                 coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
--------------
Intercept                     51.0013      0.620     82.217      0.000
49.782      52.220
ActiveUsers:InteractingPct    -2.3278      0.269     -8.662      0.000
```

```
-2.856      -1.800
ActiveUsers:OSType_MacOS       3.6143      0.780      4.633      0.000
2.082       5.147
ActiveUsers:OSType_Windows    -1.8764      0.768     -2.442      0.015
-3.386      -0.367
ActiveUsers:OSType_iOS        -2.6289      0.795     -3.306      0.001
-4.191      -1.067
ActiveUsers                   10.0047      0.584     17.124      0.000
8.857      11.153
InteractingPct                 5.0348      0.266     18.931      0.000
4.512       5.557
OSType_MacOS                   9.4142      0.757     12.438      0.000
7.927      10.901
OSType_Windows                 3.8746      0.761      5.090      0.000
2.379       5.370
OSType_iOS                    -5.7581      0.787     -7.319      0.000
-7.304      -4.212
I(ActiveUsers ** 2)           -2.6981      0.285     -9.480      0.000
-3.257      -2.139
==============================================================================
Omnibus:                      230.750   Durbin-Watson:                   1.928
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3263.977
Skew:                           1.617   Prob(JB):                         0.00
Kurtosis:                      15.066   Cond. No.                         8.06
==============================================================================

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

[12]:
```
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()
print(f"Kontrola hodnôt podla VIF \n{vif_df}")
```

```
Kontrola hodnôt podla VIF
                                 0
Intercept                 5.520433
ActiveUsers:InteractingPct  1.019165
ActiveUsers:OSType_MacOS    2.323097
ActiveUsers:OSType_Windows  2.359303
ActiveUsers:OSType_iOS      2.233026
ActiveUsers                 4.896789
InteractingPct              1.014742
OSType_MacOS                1.638202
```

```
OSType_Windows              1.626717
OSType_iOS                  1.596451
I(ActiveUsers ** 2)         1.018598
```

[13]:
```
#Identifikácia a odstránenie odlahlých hodnôt pomocou Cookovej vzdialenosti
# takéto hodnoty majú výrazný vplyv na model a zhoršujú predikciu modelu
influence = results.get_influence()
cooks_d = influence.cooks_distance[0]

#indexy najväčších Cookových vzdialeností
outl_index = np.where(cooks_d > 4 / df_encoded.shape[0])[0]
df_encoded.iloc[outl_index]

print(f"Hodnoty na odstránenie:")
print(tabulate(df_encoded.iloc[outl_index], headers='keys', tablefmt='pretty',␣
 ↪showindex=False))
```

```
Hodnoty na odstránenie:
+---------------------+---------------------+--------------------+-----------+
-------------+---------------+------------+
|     ActiveUsers     |    InteractingPct   |    ScrollingPct    | Ping [ms] |
OSType_MacOS | OSType_Windows | OSType_iOS |
+---------------------+---------------------+--------------------+-----------+
-------------+---------------+------------+
|  1.376129175561142  |  1.6658064665784056 |       0.0188       |   55.0    |
0.0      |      1.0      |    0.0     |
|  1.5029751361218042 |  1.619138291985584  | 0.0325999999999999 |   45.0    |
0.0      |      0.0      |    1.0     |
| -1.5503602922346924 | -0.8184283925438369 |       0.7534       |   13.0    |
0.0      |      0.0      |    0.0     |
|  0.9704577289383431 | -1.5705299888803286 |       0.9758       |   35.0    |
0.0      |      0.0      |    1.0     |
|  -1.229121915334935 |  1.6549848608757223 |       0.022        |   36.0    |
0.0      |      0.0      |    1.0     |
|  1.5826956624184432 | 0.17817385763765628 |       0.4587       |   60.0    |
0.0      |      0.0      |    1.0     |
|  -0.81088368624173  |  1.3411582954979058 |       0.1148       |   37.0    |
0.0      |      0.0      |    0.0     |
| -1.0410128902310427 |  1.4378763964656378 |       0.0862       |   56.0    |
0.0      |      0.0      |    1.0     |
| -1.6803479484129562 |  -1.393326195498889 |       0.9234       |   33.0    |
0.0      |      1.0      |    0.0     |
| 0.010669717761072691 | 0.008748093355020418 |       0.5088       |   90.0    |
0.0      |      1.0      |    0.0     |
|  1.4692019082325876 | -0.6804529198346245 |       0.7126       |   60.0    |
0.0      |      0.0      |    1.0     |
| -1.6803479484129562 |  1.5345944974333703 | 0.0575999999999999 |   35.0    |
1.0      |      0.0      |    0.0     |
```

```
| -1.1380127424244904  |  0.9759291030323433  |       0.2228       |   56.0    |
0.0      |      0.0       |    0.0     |
|  1.5139710707834095  |  0.5153345103118839  |       0.359        |   51.0    |
0.0      |      1.0       |    0.0     |
| -1.2102717416293258  |  1.5372998988590412  | 0.0567999999999999 |   64.0    |
1.0      |      0.0       |    0.0     |
| -2.0942663460319593  | -0.9384805808079801  |       0.7889       |   61.0    |
1.0      |      0.0       |    0.0     |
|  1.3168296707789129  | -0.13328548149269845 |       0.5508       |   37.0    |
0.0      |      0.0       |    1.0     |
+----------------------+----------------------+--------------------+-----------+
-------------+---------------+------------+
```

[14]:
```
#odstránenie odlahlých hodnôt
new_df_encoded = df_encoded.drop(outl_index, axis=0)

new_all_combinations = create_combination_multiplication(new_df_encoded)
model = smf.ols(formula=formula, data=new_all_combinations)
new_results = model.fit()
```

[15]:
```
#2.2 Identifikácia problematických hodnôt
max_ping = new_results.predict().argmax()

print(f"Najaväčšia hodnota odozvy: {df2['Ping [ms]'][max_ping]}\n")
print(f"nastavenie parametrov:\n{df2.loc[max_ping]}")
```

```
Najaväčšia hodnota odozvy: 72

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

[16]:
```
#2.3 Odhad hodnoty odozvy pre Windows užívateľa
windows_user = {'ActiveUsers': new_all_combinations['ActiveUsers'].
 ↪mean(),'InteractingPct': new_all_combinations['InteractingPct'].mean(),
             'ActiveUsers:InteractingPct': new_all_combinations['ActiveUsers:
 ↪InteractingPct'].mean(),
             'OSType_Windows':1, 'OSType_MacOS':0, 'OSType_iOS':0}

tmp = results.get_prediction(pd.DataFrame(windows_user, index=[0])).conf_int()

print("Odhadnutá hodnota odozvy:", results.predict(pd.DataFrame(windows_user,␣
 ↪index=[0]))[0])
```

```
print("Konfidenčný interval:", pd.DataFrame(tmp, columns=['lower', 'upper']).
  ↪iloc[0])
print("Predikčný interval:", results.get_prediction(pd.DataFrame(windows_user,␣
  ↪index=[0])).conf_int(obs=True)[0])
```

```
Odhadnutá hodnota odozvy: 54.849133808989585
Konfidenčný interval: lower    53.687713
upper    56.010555
Name: 0, dtype: float64
Predikčný interval: [43.16861145 66.52965616]
```

2.4

Koeficient determinácie naznačuje, že priamka vystihujé dáta približne na 89%.

R-squared: 0.890

Hodnota F-statistic s nízkou p hodnotu naznačuje, že alespoň jedna z prediktorových premenných významne súvisí s odpovedajúcimi premennými.Ide o významný keoficient ktorý pomáha modelovať y v tomto prípade ping

F-statistic: 384.2,

Durbin-Watsonova hodnota je blízko k číslu 2 čo naznačuje žiadnu významnú autokorelacu.

Durbin-Watson: 1.908,

p-hodnoty (P>|t|) označujú štatistickú významnosť každého koeficientu. Nízka p-hodnota u všetkých premenných naznačuje, že zodpovedajúce premenné sú štatisticky významné. Kedže všetky premenné majú 0,0 hodnotu tak sú všetky významné.

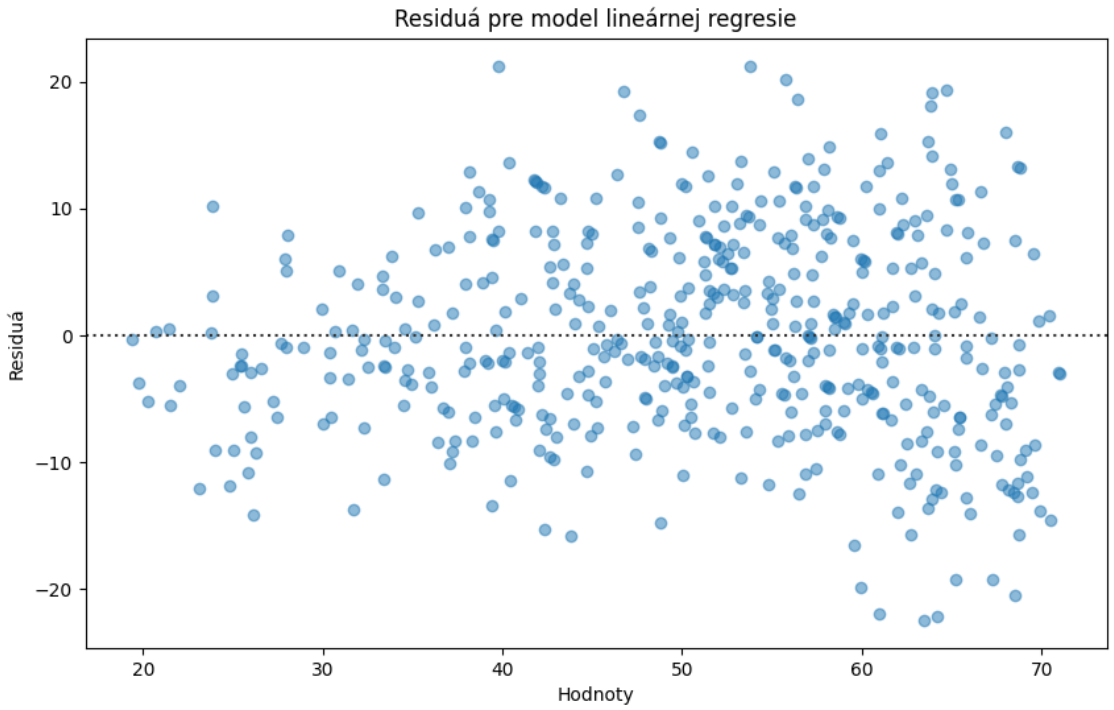
Na základe týchto hodnôt sa dá povedať, že model je primerane vhodný na ďalšie použitie.

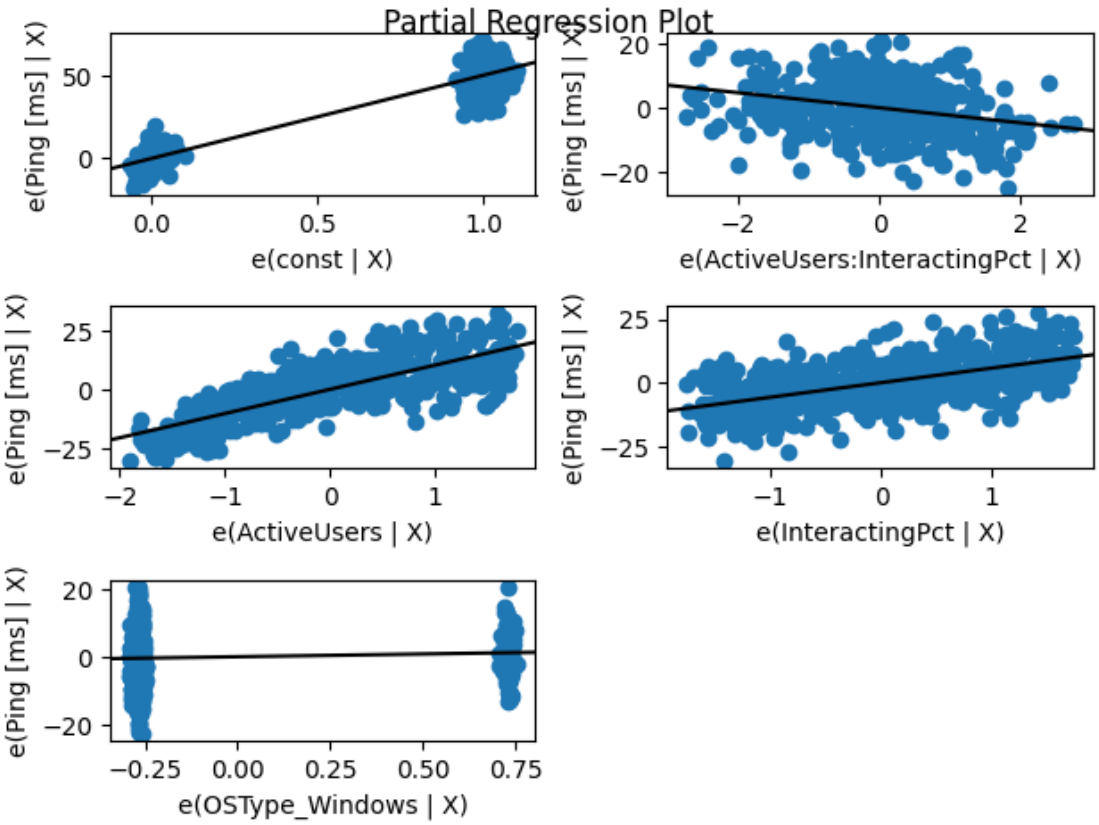
Následné grafy taktiež ukazujú použitelnsoť tohto modelu.

[17]:
```
X = new_all_combinations[['ActiveUsers:InteractingPct', 'ActiveUsers',␣
  ↪'InteractingPct', 'OSType_Windows']]
y = new_all_combinations['Ping [ms]']
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()

print("Čím je väčšie náhodné rozhádzanie hodnôt v okolí čiar tím je väčšia␣
  ↪šanca,\n\
že sa nenachádza žiadna systematická chyba a model je vhodný pre dané dáta.\n")

plt.figure(figsize=(10, 6))
sns.residplot(x=model.fittedvalues, y=model.resid, scatter_kws={'alpha': 0.5})
plt.title('Residuá pre model lineárnej regresie')
plt.xlabel('Hodnoty')
plt.ylabel('Residuá')
plt.show()
```

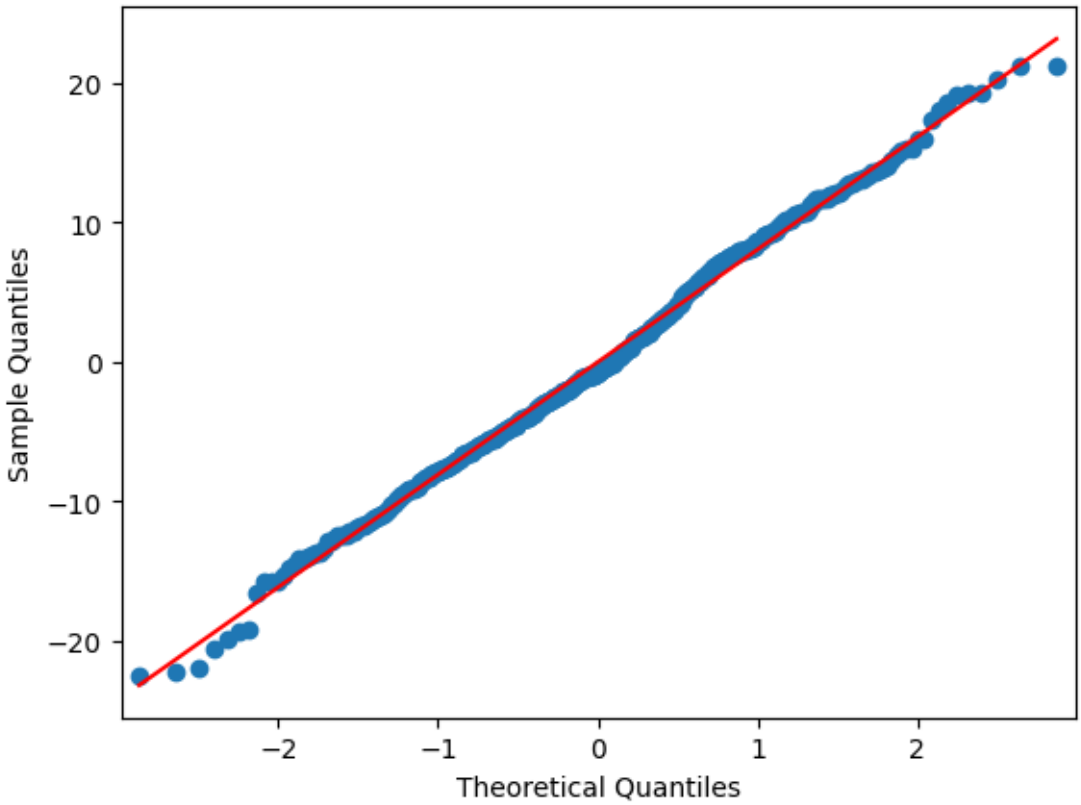
```
print("\n")
# Kontrola linearity - Grafové čiastočné regresie
sm.graphics.plot_partregress_grid(model)
plt.show()

# Nezávislosť reziduí - Durbin-Watson štatistika
durbin_watson_statistic = sm.stats.stattools.durbin_watson(model.resid)
print(f'Durbin-Watson štatistika: {durbin_watson_statistic}')

# Normálne rozloženie reziduí - Q-Q graf
sm.qqplot(model.resid, line='s')
plt.show()
```

Čím je väčšie náhodné rozhádzanie hodnôt v okolí čiar tím je väčšia šanca, že sa nenachádza žiadna systematická chyba a model je vhodný pre dané dáta.





Durbin-Watson štatistika: 1.8929631337668966


Sample Quantiles
Theoretical Quantiles
20
10
0
−10
−20
−2
−1
0
1
2