# Analiza skupa podataka Food choices

Tijana Jevtić
Jelena Mrdak

21. juni 2018

**Sažetak**

Skup podataka Food choices predstavlja skup odgovora studenata sa koledza na odredjena pitanja vezana za njihove navike u ishrani. Dakle, podaci su dobijeni anketiranjem 126 studenata na 60 pitanja razlicitog tipa. Neka pitanja se konkretno ticu njihovih preferenca vezanih za hranu, njihovih navika, dok se kroz ostala dobija vise informacija o njihovom socio-ekonomskom statusu. Sve to zajedno cini bogat skup podataka iz kog se moze zakljuciti dosta zanimljivih pravilnosti.

Koliko se današnji studenti brinu o ishrani i koliko znaju o hrani? Da li njihove navike stečene još u detinjstvu utiču na to kako se danas hrane? Koliko njihove kulinarske veštine utiču na način na koji se hrane? Zasto jedu i da li je jedini odgovor zbog gladi? Da li odgovor na prethodno pitanje zavisi od pola?

Na ova i još mnogobrojna pitanja, pokušaćemo da odgovorimo u ovom radu.

# Sadržaj



# 1 Opis skupa podataka

Skup podataka - datoteka food_coded.csv sadrzi sadrzi 60 razlicitih atributa. Podaci su, za sada, neocisceni. U nastavku cemo pomenuti svaki od atributa kako bi se citalac upoznao sa skupom podataka, a posebnu paznju cemo posvetiti onima koje koristimo u istrazivanju.

- GPA - numerički, prosek na fakultetu

- Pol - kategorički
  **1** - žensko
  **2** - muško

- Doručak - koja od 2 ponudjene slike ispitanike vise asocira na dorucak
  Ispitanicima je ponuđena slika pahuljica i krofne i treba da kažu šta ih asocira na doručak
  **1** - pahuljice
  **2** - krofna

- Procena kalorija u jednom parčetu piletine
  **1** - 265
  **2** - 430
  **3** - 610
  **4** - 720

- Da li je bitna količina kalorija koja se konzumira dnevno
  **1** - ne znam koliko kalorija treba konzumirati dnevno
  **2** - uopšte nije bitno
  **3** - umereno je bitno
  **4** - veoma je bitno

- Procena kalorija u kolacu iz Starbucks-a
  **1** - 107 cal
  **2** - 315 cal
  **3** - 420 cal
  **4** - 980 cal

- Kafa - koja od 2 ponudjene slike ispitanike vise asocira na kafu

- Hrana za utehu
  - koja hrana asocira ispitanike na dom i lepe uspomene, cini ih srecnim Ispitanici treba da navedu izmedju 3 i 5 razlicitih jela.

- Zasto posezu za hranom za utehu
  Ispitanici treba da navedu do 3 razloga zašto jedu hranu za utehu? (npr. tuga, sreća, bes)

- Hrana za utehu - kodirana
  **1** - stres
  **2** - dosada
  **3** - depresija
  **4** - glad
  **5** - lenjost
  **6** - hladno vreme

**7** - sreća
**8** - gledanje televizije
**9** - ništa od navedenog

- Kuvanje - koliko ispitanici cesto kuvaju
  **1** - svakog dana
  **2** - nekoliko dana nedeljno
  **3** - koliko cesto mogu, ali retko
  **4** - jedino za praznike
  **5** - nikada

- Koji tip kuhinje su jeli kad su bili mali
  **1** - americka
  **2** - meksicka/spanska
  **3** - korejska/azijska
  **4** - indijska
  **5** - americka sa internacionalnim jelima
  **6** - ostalo

- Trenutna dijeta

- Trenutna dijeta - kodirano od 1 do 4

- Koju sliku asociraju sa picem

- Kako se ishrana promenila od kada su na koledzu

- Kako se ishrana promenila
  **1** - na losija
  **2** - na bolje
  **3** - isto
  **4** - nije sigurno

- Kako se ishrana promenila u terminima toga sta jedu

- Prejedanje - koliko se ispitanici cesto prejedaju tokom nedelje
  **1** - nikad
  **2** - 1 do 2 puta
  **3** - 2 do 3 puta

**4** - 3 do 5 puta
**5** - svaki dan

- Zaposlenje - da li su zaposleni **1** - da, puno radno vreme
  **2** - da, poluradno vreme
  **3** - ne
  **4** - drugo

-

- Vezbanje - koliko puta nedeljno vezbaju
  **1** - svakodnevno
  **2** - 2 do 3 puta
  **3** - jednom
  **4** - ponekad
  **5** - nikad

- Ocevo obrazovanje

- Oceva profesija

- Omiljena vrsta kuhinje

- Omiljena vrsta kuhinje - kodirano

- Omiljena hrana - kuvana ili kupljena
  **1** - kuvana kod kuce
  **2** - kupljena u prodavnici
  **3** - oba

- Omiljena hrana iz detinjstva

- Koja od 2 slike ih vise asocira sa krompiricima

- Voce - koliko je verovatno da ce pojesti voce tokom dana
  **1** - vrlo verovatno
  **2** - ne toliko verovatno
  **3** - onako
  **4** - verovatno

**5** - vrlo verovatno

- Godina na koledzu

- Grcka hrana - koliko je verovatno da bi je jeli

- Koliko se zdravo osecaju od 1 do 10

- Sta je, po njihovim recima, zdrav obrok

- Sta je, po njihovim recima, idealna ishrana

- Idealna ishrana - kodirano

- Zarada - koliko zaradjuju godisnje
  **1** - manje od 15 000 dolara
  **2** - 15 001 - 30 000 dolara
  **3** - 30 001 - 50 000 dolara
  **4** - 50 001 - 70 000 dolara
  **5** - 70 001 - 100 000 dolara
  **6** - vise od 100 000 dolara

- Indijska hrana - koliko je verovatno da bi je jeli

- Italijanska hrana - koliko je verovatno da bi je jeli

- Koliko je zivot lep od 1 do 10

- Bracni zivot
  **1** - slobodan/slobodna
  **2** - u vezi
  **3** - zivi sa partnerom
  **4** - udat/ozenjen
  **5** - razveden/a
  **6** - udovica

- Sta bi posluzili kao veceru prijatelju

- Majcino obrazovanje

- Majcina profesija

- Koliko cesto proveravaju kolicinu kalorija hrane koje konzumiraju

- Da li zive ili ne na kampusu

- Koliko puta nedeljno roditelji kuvaju
  **1** - skoro svaki dan
  **2** - 2 do 3 puta
  **3** - 1 do 2 puta
  **4** - za vreme odmora
  **5** - nikad

- Placanje obroka - koliko bi novca potrosili na obrok

- Persijska hrana - koliko je verovatno da bi je jeli

- Procena sopstvene tezine
  **1** - vitak/vitka
  **2** - vrlo utreniran/utrenirana
  **3** - tacno kako treba
  **4** - pomalo debeo/debela
  **5** - debeo/debela
  **6** - ne razmisljam o tome kada mislim o sebi

- Koja od 2 slike ih vise asocira sa supom

- Sport - da li se bave sportom
  **1** - da
  **2** - ne
  **99** - bez odgovora

- Tajlandska hrana - koliko je verovatno da bi je jeli

- Koliko tortilja ima kalorija
- Koliko guska ima kalorija
- Sport - kojim sportom se bave
- Povrce - koliko je verovatno da jedu povrce na dnevnoj bazi
- Vitamini - da li uzimaju dodatne vitamine
- Koliko vafl ima kalorija
- Tezina (u funtama: 1kg = 2.20462 funti)

# 2 Primer klasifikacije

Odredićemo klasifikaciju na osnovu atributa comfort_food_reasons_coded, cook i eating_out, dok će nam ciljni atribut biti weight.

Najpre ćemo učitati podatke i prikazati prvih pet redova.

```
df = pd.read_csv('./food-choices/food_coded.csv')
print('\n{}'.format(df.head()))
```

```
  GPA  Gender  breakfast ... waffle_calories                    weight
0   2.4       2          1 ...            1315                       187
1 3.654       1          1 ...             900                       155
2   3.3       1          1 ...             900  I'm not answering this.
3   3.2       1          1 ...            1315            Not sure, 240
4   3.5       1          1 ...             760                       190
```

Možemo uraditi osnovnu statistiku za svaku kolonu.

```
print("\nStatistike skupa:\n{}".format(df.describe()))
```

```
Statistike skupa:
           Gender   breakfast  calories_chicken ... veggies_day    vitamins
count  125.000000  125.000000        125.000000 ...  125.000000  125.000000
mean     1.392000    1.112000        577.320000 ...    4.008000    1.512000
```

```
std      0.490161    0.316636      131.214156 ...    1.081337    0.501867
min      1.000000    1.000000      265.000000 ...    1.000000    1.000000
25%      1.000000    1.000000      430.000000 ...    3.000000    1.000000
50%      1.000000    1.000000      610.000000 ...    4.000000    2.000000
75%      2.000000    1.000000      720.000000 ...    5.000000    2.000000
max      2.000000    2.000000      720.000000 ...    5.000000    2.000000
```

Za algoritam koji želimo da primenimo, izdvojićemo sledeće atribute: comfort_food_reasons_coded, cook, eating_out i weight.

```
target_attribute = 'weight'
attribute_1 = 'comfort_food_reasons_coded'
attribute_2 = 'cook'
attribute_3 = 'eating_out'

df = df[[attribute_1, attribute_2, attribute_3, target_attribute]]
```

```
   comfort_food_reasons_coded  cook  eating_out                    weight
0                         9.0   2.0           3                       187
1                         1.0   3.0           2                       155
2                         1.0   1.0           2  I'm not answering this.
3                         2.0   2.0           2            Not sure, 240
4                         1.0   1.0           2                       190
```

Kao što možemo primetiti, nisu sve vrednosti celobrojne. Zato ćemo obrisati sve redove koji sadrže NaN-ove ili stringove u nekoj od ove četiri kolone. Takođe, vrednosti u koloni weight ćemo transformisati. Preslikaćemo ih u skup $\{0, 1, 2\}$.

```
df = df.replace('nan', np.nan)
df = df.dropna()


df = df[df[target_attribute].apply(lambda x: str(x).isdigit())]

df.reset_index(drop=True, inplace=True)

df[attribute_1] = df.comfort_food_reasons_coded.astype(int)
df[attribute_2] = df.cook.astype(int)
df[attribute_3] = df.eating_out.astype(int)
df[target_attribute] = df.weight.astype(int)

changes = {}
weight = df[target_attribute].unique()
for w in weight:
    if int(w) <150:
        changes[w] = 0
    elif int(w) < 190:
        changes[w] = 1
```

```
        else:
            changes[w] = 2

    df[target_attribute] = df[target_attribute].replace(changes)
```

```
   comfort_food_reasons_coded  cook  eating_out  weight
0                           9     2           3       1
1                           1     3           2       1
2                           1     1           2       2
3                           4     3           1       2
4                           1     2           2       1
```

Sada ćemo izvršiti podelu skupa na test i trening skup.

```
    X = df[[attribute_1, attribute_2, attribute_3]]
    y = df[[target_attribute]]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    print("\nVelicina skupa za obucavanje: {}".format(X_train.size))
    print("Velicina skupa za testiranje: {}".format(X_test.size))
```

```
Velicina skupa za obucavanje: 207
Velicina skupa za testiranje: 90
```

Pošto smo izvršili podelu skupa, primenićemo algoritam za klasifikaciju - $k$ najbližih suseda.

```
    clf = KNeighborsClassifier(5, 'distance')

    # Treniramo model
    clf.fit(X_train, y_train.values.ravel())

    # Vrsimo predikciju
    y_test_predicted = clf.predict(X_test)
    y_train_predicted = clf.predict(X_train)

    # Izracunavamo preciznost
    train_acc = clf.score(X_train, y_train)
    test_acc = clf.score(X_test, y_test)
    print('train preciznost: {}'.format(train_acc))
    print('test preciznost: {}'.format(test_acc))
```

```
train preciznost: 0.7536231884057971
test preciznost: 0.7
```

Izveštaj i matricu konfuzije možemo dobiti na sledeći način:

```
    test_rep = sklearn.metrics.classification_report(y_test, y_test_predicted)
    train_rep = sklearn.metrics.classification_report(y_train, y_train_predicted)
    print("\nTest izvestaj:\n{}".format(test_rep))
    print("Trening izvestaj:\n{}".format(train_rep))

    train_conf = sklearn.metrics.confusion_matrix(y_train, y_train_predicted)
    test_conf = sklearn.metrics.confusion_matrix(y_test, y_test_predicted)
    print("Matrica konfuzije za skup za obucavanje:\n{}".format(train_conf))
    print("\nMatrica konfuzije za skup za testiranje:\n{}".format(test_conf))
```

```
Test izvestaj:
             precision    recall  f1-score   support

          0       0.67      0.91      0.77        11
          1       0.77      0.67      0.71        15
          2       0.50      0.25      0.33         4

avg / total       0.70      0.70      0.68        30

Trening izvestaj:
             precision    recall  f1-score   support

          0       0.74      0.86      0.79        29
          1       0.73      0.81      0.77        27
          2       1.00      0.38      0.56        13

avg / total       0.78      0.75      0.74        69

Matrica konfuzije za skup za obucavanje:
[[25  4  0]
 [ 5 22  0]
 [ 4  4  5]]

Matrica konfuzije za skup za testiranje:
[[10  1  0]
 [ 4 10  1]
 [ 1  2  1]]
```

# 3 Vizuelizacija podataka

Atributi koje posmatramo:

- self_perception_weight
  procena sopstvene težine

  1. i dont think myself in these terms

2. overweight
3. slightly overweight
4. just right
5. very fit
6. slim

- fruit_day
  dnevna konzumacija voća

1. very unlikely
2. unlikely
3. neutral
4. likely
5. very likely

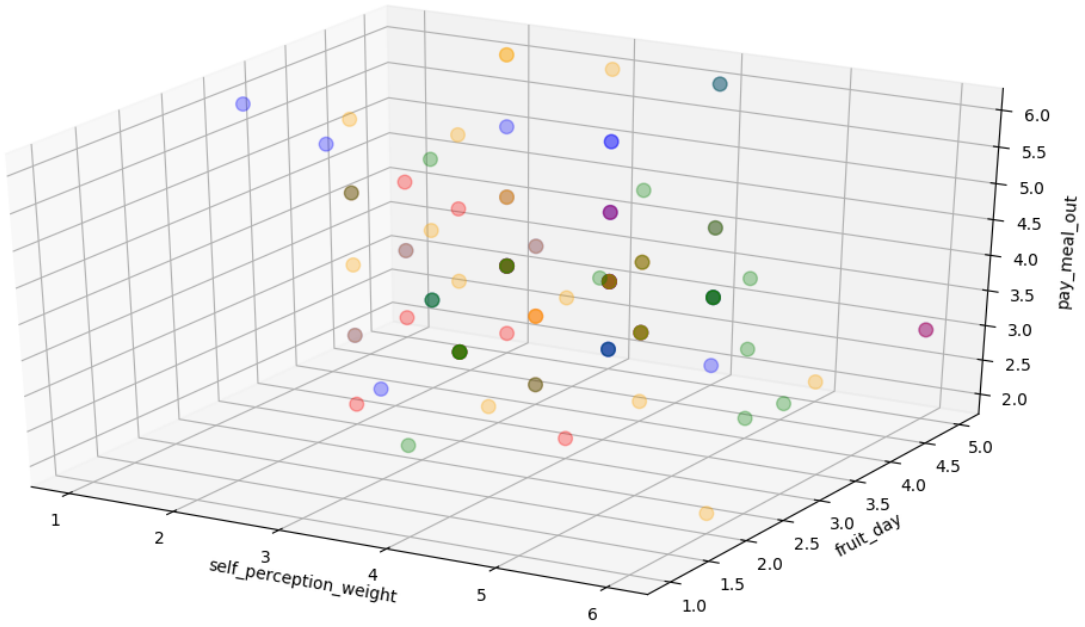
- pay_meal_out
  Koliko biste platili za jedan obrok?

1. up to 5.00
2. 5.01 to 10.00
3. 10.01 to 20.00
4. 20.01 to 30.00
5. 30.01 to 40.00
6. more than 40.01

- weight
  Unesite svoju težinu.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# read data
df = pd.read_csv('./food-choices/food_coded.csv')

# choose attributes
target_attribute = 'weight'
attribute_1 = 'self_perception_weight'
attribute_2 = 'fruit_day'
attribute_3 = 'pay_meal_out'

df = df[[attribute_1, attribute_2, attribute_3, target_attribute]]
```

```
# data preprocessing

# remove NaN
df = df.replace('nan', np.nan)
df = df.dropna()

df = df[df[target_attribute].apply(lambda x: str(x).isdigit())]

df.reset_index(drop=True, inplace=True)

df[attribute_1] = df.self_perception_weight.astype(int)
df[attribute_2] = df.fruit_day.astype(int)
df[attribute_3] = df.pay_meal_out.astype(int)
df[target_attribute] = df.weight.astype(int)

# transform target attribute
changes = {}
weight = df[target_attribute].unique()
for w in weight:
    if int(w) <=128:
        changes[w] = 0
    elif int(w) <= 155:
        changes[w] = 1
    elif int(w) <= 180:
        changes[w] = 2
    else:
        changes[w] = 3

df[target_attribute] = df[target_attribute].replace(changes)
weight = df[target_attribute].unique()

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

colors = ['green','blue','orange','red']
for (v, color) in zip(weight, colors):
    subsamples = df.loc[df[target_attribute] == v]
    ax.scatter(subsamples[attribute_1],
               subsamples[attribute_2],
               subsamples[attribute_3],
               color=color,s=70,alpha=0.3)

ax.set_xlabel('self_perception_weight')
ax.set_ylabel('fruit_day')
ax.set_zlabel('pay_meal_out')
plt.show()
```

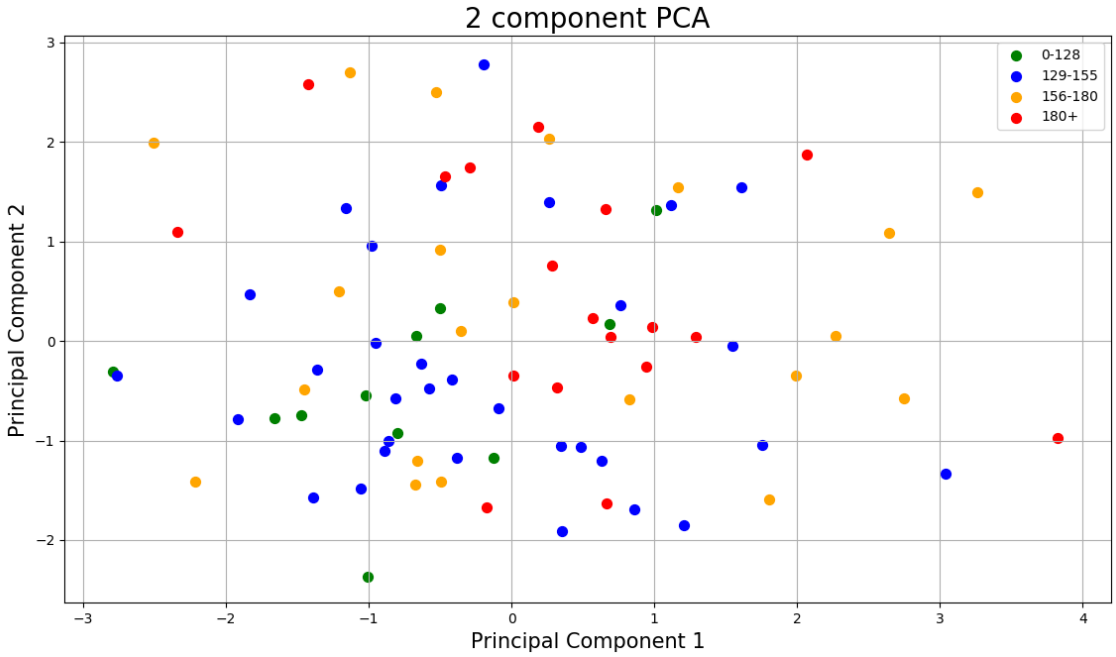



## 3.1 PCA

```
from sklearn.decomposition import PCA
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler

def isFloat(value):
  try:
    float(value)
    return True
  except ValueError:
    return False

# read data
df = pd.read_csv('./food-choices/food_coded.csv')

# choose attributes
target_attribute = 'weight'
attributes = [ 'exercise','Gender','eating_out',
               'GPA','employment','breakfast',
               'calories_chicken','calories_day','coffee',
               'diet_current_coded','drink','cook']

df = df[[attributes[0],attributes[1],attributes[2],
         attributes[3],attributes[4],attributes[5],
         attributes[6],attributes[7],attributes[8],
```

```
        attributes[9], attributes[10], attributes[11],
        target_attribute]]

# data preprocessing

# remove NaN values
df = df.replace('nan', np.nan)
df = df.dropna()

# clean data
df = df[df[target_attribute].apply(lambda x: str(x).isdigit())]
df = df[df['GPA'].apply(lambda x: isFloat(str(x)))]

df.reset_index(drop=True, inplace=True)

# convert values to numbers
df[attributes[0]] = df.exercise.astype(int)
df[attributes[1]] = df.Gender.astype(int)
df[attributes[2]] = df.eating_out.astype(int)
df[attributes[3]] = df.GPA.astype(float)
df[attributes[4]] = df.employment.astype(int)
df[attributes[5]] = df.breakfast.astype(int)
df[attributes[6]] = df.calories_chicken.astype(int)
df[attributes[7]] = df.calories_day.astype(int)
df[attributes[8]] = df.coffee.astype(int)
df[attributes[9]] = df.diet_current_coded.astype(int)
df[attributes[10]] = df.drink.astype(int)
df[attributes[11]] = df.cook.astype(int)
df[target_attribute] = df.weight.astype(int)

# transform target attribute
changes = {}
weight = df[target_attribute].unique()

for w in weight:
    if int(w) <=128:
        changes[w] = 0
    elif int(w) <= 155:
        changes[w] = 1
    elif int(w) <= 180:
        changes[w] = 2
    else:
        changes[w] = 3

df[target_attribute] = df[target_attribute].replace(changes)
weight = df[target_attribute].unique()
```

```
features = attributes

# extract features
x = df.loc[:, features].values
# extract target attribute
y = df.loc[:,[target_attribute]].values

x = StandardScaler().fit_transform(x)

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             ,columns=['principal component 1','principal component 2'])

finalDf = pd.concat([principalDf,df[[target_attribute]]],axis = 1)

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1',fontsize = 15)
ax.set_ylabel('Principal Component 2',fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

targets = [0,1,2,3]
colors = ['green','blue','orange','red']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf[target_attribute] == target
    ax.scatter(finalDf.loc[indicesToKeep,'principal component 1']
               ,finalDf.loc[indicesToKeep,'principal component 2']
               ,c=color
               ,s=50)

ax.legend(['0-128', '129-155','156-180','180+'])
ax.grid()
plt.show()
```




# Literatura

[1] Knjiga1

[2] Knjiga2