Analiza skupa podataka Food choices

Tijana Jevtić
Jelena Mrdak

28. juni 2018

Sažetak

Skup podataka Food choices predstavlja skup odgovora studenata sa koledža na određena pitanja vezana za njihove navike u ishrani. Dakle, podaci su dobijeni anketiranjem 125 studenata na 61 pitanje različitog tipa. Neka pitanja se konkretno tiču njihovih preferenca vezanih za hranu, njihovih navika, dok se kroz ostala dobija više informacija o njihovom socio-ekonomskom statusu. Sve to zajedno čini bogat skup podataka iz kog se moze zaključiti dosta zanimljivih pravilnosti.

Šta današnji studenti na koledžu najviše vole da jedu? Koliko često kuvaju? Zašto jedu i da li je jedini odgovor zbog gladi? Koliko to što konzumiraju prekomerne količine hrane i razlozi zbog kojih konzumiraju svoju omiljenu hranu utiču na njihovu težinu?

Na ova i još mnogobrojna pitanja, pokušaćemo da odgovorimo u ovom radu koristeći programski jezik python i adekvatne biblioteke, kao i alat Knime.

Sadržaj

1 Opis skupa podataka

Za početak ćemo se površno upoznati sa skupom podataka koji se nalazi u datoteci food_coded.csv.

```
df = pd.read_csv('./food_coded.csv', na_values="none")
print(data.shape)
```

Skup podataka je učitan, nedostajuće vrednosti su obeležene sa none i izlaz je:

```
(125, 61)
```

Skup se sastoji od 61 atributa i 125 redova.
Atribute dobijamo:

```
print(data.columns)
```

Spisak svih atributa:

```
Index(['GPA', 'Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'comfort_food', 'comfort_food_reasons',
       'comfort_food_reasons_coded', 'cook', 'comfort_food_reasons_coded.1',
       'cuisine', 'diet_current', 'diet_current_coded', 'drink',
       'eating_changes', 'eating_changes_coded', 'eating_changes_coded1',
       'eating_out', 'employment', 'ethnic_food', 'exercise',
       'father_education', 'father_profession', 'fav_cuisine',
       'fav_cuisine_coded', 'fav_food', 'food_childhood', 'fries', 'fruit_day',
       'grade_level', 'greek_food', 'healthy_feeling', 'healthy_meal',
       'ideal_diet', 'ideal_diet_coded', 'income', 'indian_food',
       'italian_food', 'life_rewarding', 'marital_status',
       'meals_dinner_friend', 'mother_education', 'mother_profession',
       'nutritional_check', 'on_off_campus', 'parents_cook', 'pay_meal_out',
       'persian_food', 'self_perception_weight', 'soup', 'sports', 'thai_food',
       'tortilla_calories', 'turkey_calories', 'type_sports', 'veggies_day',
       'vitamins', 'waffle_calories', 'weight'],
      dtype='object')
```

U nastavku ćemo pomenuti svaki od atributa kako bi se čitalac upoznao sa skupom podataka, a posebnu pažnju ćemo posvetiti onima koje koristimo u istraživanju.

- GPA - numerički, prosek na fakultetu
- Pol - kategorički
  1 - žensko
  2 - muško
- Doručak - koja od 2 ponudjene slike ispitanike više asocira na doručak

- 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 kolaču iz Starbucks-a
  1 - 107 cal
  2 - 315 cal
  3 - 420 cal
  4 - 980 cal

- Kafa - koja od 2 ponuđene slike ispitanike više asocira na kafu

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

- Zašto posežu 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 često mogu, ali retko
  4 - jedino za praznike
  5 - nikada

- Koji tip kuhinje su jeli kad su bili mali
  1 - američka
  2 - meksička/španska

3 - korejska/azijska
4 - indijska
5 - američka 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 lošije
  2 - na bolje
  3 - isto
  4 - nije sigurno
- Kako se ishrana promenila u terminima toga šta jedu
- Konzumiranje prekomernih količina hrane - koliko često to rade 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
- Lokalna hrana - koliko je verovatno da bi jeli hranu sa nekom područja
- Vežbanje - koliko puta nedeljno vežbaju
  1 - svakodnevno
  2 - 2 do 3 puta
  3 - jednom
  4 - ponekad
  5 - nikad
- Očevo obrazovanje
- Očeva profesija
- Omiljena vrsta kuhinje
- Omiljena vrsta kuhinje - kodirano

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

- Omiljena hrana iz detinjstva

- Koja od 2 slike ih više asocira sa krompirićima

- Voće - koliko je verovatno da će pojesti voće tokom dana
  1 - vrlo verovatno
  2 - ne toliko verovatno
  3 - onako
  4 - verovatno
  5 - vrlo verovatno

- Godina na koledzu

- Grčka hrana - koliko je verovatno da bi je jeli

- Koliko se zdravo osećaju od 1 do 10

- Šta je, po njihovim rečima, zdrav obrok

- Šta je, po njihovim rečima, idealna ishrana

- Idealna ishrana - kodirano

- Zarada - koliko zarađuju godišnje
  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 - više od 100 000 dolara

- Indijska hrana - koliko je verovatno da bi je jeli

- Italijanska hrana - koliko je verovatno da bi je jeli

- Koliko je život lep od 1 do 10

- Bračni zivot
  1 - slobodan/slobodna
  2 - u vezi
  3 - živi sa partnerom
  4 - udat/oženjen
  5 - razveden/a
  6 - udovica

- Šta bi poslužili kao večeru prijatelju
- Majčino obrazovanje
- Majčina profesija
- Koliko često proveravaju količinu kalorija hrane koje konzumiraju
- Da li žive 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
- Plaćanje obroka - koliko bi novca potrošili na obrok
- Persijska hrana - koliko je verovatno da bi je jeli
- Procena sopstvene težine
  1 - vitak/vitka
  2 - vrlo utreniran/utrenirana
  3 - tačno kako treba
  4 - pomalo debeo/debela
  5 - debeo/debela
  6 - ne razmišljam o tome kada mislim o sebi
- Koja od 2 slike ih visě 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
- Povrće - koliko je verovatno da jedu povrće na dnevnoj bazi
- Vitamini - da li uzimaju dodatne vitamine
- Koliko vafl ima kalorija
- Težina (u funtama: 1kg = 2.20462 funti)

2 Detaljnije upoznavanje sa skupom podataka

Pogledajmo koliko ima ispitanika kog pola:

```
# kolonu Gender prebacujemo u brojevni tip i stampamo dobijene tabele
# zenski pol je kodiran brojem 1, a muski brojem 2

data.Gender = data.Gender.astype(int)
print(data[data['Gender'] == 1])
print(data[data['Gender'] == 2])
```

Nakon prvog ispisa dobijamo tabelu koja je sačinjena samo od ispitanika koji su ženskog pola i prvih nekoliko redova izgleda ovako:

	GPA	Gender	...	waffle_calories	weight
1	3.654	1	...	900	155
2	3.3	1	..	900	I'm not answering this.
3	3.2	1	...	1315	Not sure, 240
4	3.5	1	...	760	190
5	2.25	1	...	1315	190
7	3.3	1	...	1315	137
8	3.3	1	...	760	180

Nakon drugog ispisa dobijamo tabelu koja je sačinjena samo od ispitanika koji su muškog pola i prvih nekoliko redova izgleda ovako:

	GPA	Gender	breakfast	...	vitamins	waffle_calories	weight
0	2.4	2	1	...	1	1315	187
6	3.8	2	1	...	1	1315	180
12	3.4	2	1	...	2	575	264
14	3.1	2	1	...	1	900	185
15	NaN	2	2	...	2	1315	180
17	3.6	2	1	...	2	900	170

Ukupno ima 76 ispitanika koji su ženskog pola, 46 koji su muškog.

Već nakon ovako jednostavnog manipulisanja podacima možemo zaključiti nešto:
na osnovu podataka koje imamo - muškarci nemaju problem da kažu koliko su teški, što se može videti iz toga da postoje podaci za skoro sve muške osobe kada je kolona weight u pitanju. Kada se pogleda kolona weight za ženske ispitanike vidi se da postoje neke od njih koje imaju problem da kažu svoju težinu.

Podatke ćemo grupisati po polu i onda izračunati mean za svaki od atributa i možda zaključiti još nešto.

```
# pravi se novi skup gde su podaci grupisani na osnovu pola ispitanika
# u nasem slucaju ce biti grupisani u 2 reda
```

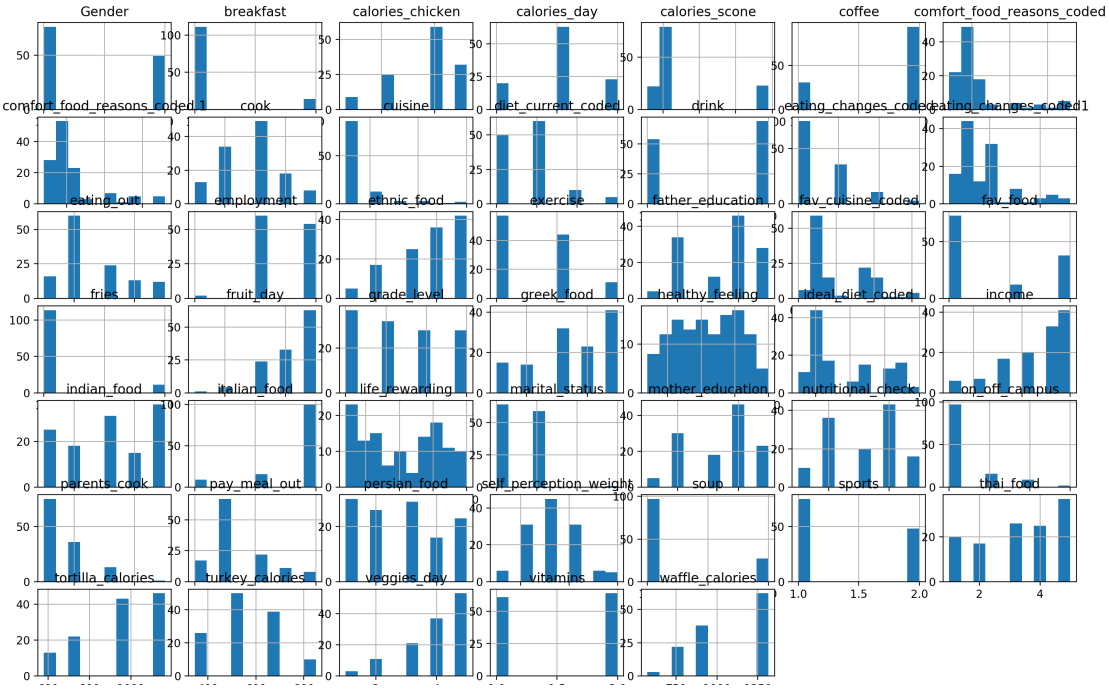
```
groupby_gender = data.groupby('Gender')

# kako nije moguce videti ceo skup,
# skup se mora odstampati na poseban nacin

with pd.option_context('display.max_rows', \
  None, 'display.max_columns', 100):
  print(groupby_gender.mean())
```

Izdvojene su neke, nama interesantne, kolone:

```
Gender  comfort_food_reasons_coded      cook
1                         2.419355  2.527027
2                         3.090909  3.187500

Gender  eating_out  exercise   healthy_feeling
1         2.552632  1.661538          5.328947
2         2.571429  1.489362          5.653061

Gender   self_perception_weight    veggies_day
1        3.320000                    4.131579
2        2.816327                    3.816327
```

Na osnovu srednje vrednosti svake kolone posebno za muškarce i posebno za žene vidimo da, u proseku, žene svoju omiljenu hranu konzumiraju iz dosade i depresivnog raspoloženja, dok muškarci najviše zbog depresivnog rasploženja.
Sa druge strane, potpuno očekivano, žene kuvaju što česće mogu. Za muškarce se ne može ništa zaključiti na osnovu srednje vrednosti ovog atributa koja je 3.18 i tačno između odgovora da kuvaju kad god mogu i da kuvaju jako retko.
Pripadnici oba pola prosečno konzumiraju prekomerne količine hrane i vežbaju jednako. Na pitanje koliko se zdravo osećaju su odgovorili otprilike približno.
Na to kako procenjuju svoju teĭinu, većina muškaraca je odgovorila da se oseća utrenirano, dok su žene ponovo malo nesigurnije - odgovori su između "otprilike kako treba"i "malo ugojeno". U isto vreme, šansa da žene jedu povrće dnevno je za nijansu veća od onih koju procenjujemo za muške ispitanike.

```
# korisno je i pogledati skup
# nad citavim skupom, tacnije, za svaku kolonu
# napravice se histogram

data.hist()
plt.show()
```



Na osnovu histograma se jednostavno zaključuje da ima mnogo više žena, na primer. Za pitanja - atribute, koja su oblika "Od ponuđene dve slike izabrati...", lako se može dobiti odgovor šta su ljudi odgovarali u proseku.
Tako, velika većina njih je na pitanje šta za njih predstavlja doručak odgovorila sa "žitarice", velika većina njih misli da je umereno važno koliko se kalorija unosi dnevno. Veliki broj ispitanika svoju omiljenu hranu jede iz dosade, zbog stresa i depresije, dok zanemarljiv broj njih jede zbog gladi (što bi trebalo da je glavni razlog zbog koga konzumiramo hranu). Takođe, većina njih se bavi fizičkom aktivnošću. Na osnovu nekoliko histograma, može se videti da je veliki broj ispitanika ljubitelj italijanske hrane: paste, testenina, pice, dok jako veliki broj njih ne bi okusilo indijsku hranu.

3 Klasifikacija

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

```
# najpre cemo ucitati 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 niske u nekoj od ove četiri kolone. Takođe, vrednosti u koloni weight ćemo transformisati. Preslikaćemo ih u skup $\{0, 1, 2\}$. Dakle, želimo da klasifikujemo ispitanike po težini, u 3 grupe, u zavisnosti od toga koliko često konzumiraju prekomerne količine hrane, koliko često kuvaju i zašto jedu hranu za utehu. U prvu grupu spadaju oni koji su teški do 150 funti (oko 70kg), u drugu oni koji imaju između 70kg i 90 kg i u treću grupu oni sa preko 90kg.

Prvo se brišu nedostajuće vrednosti.

```
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))
```

3.1 Klasifikacija korišćenjem algoritma K najbližih suseda

```
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]]
```

Kao što se može videti iz matrica konfuzije za test i trening skup, podaci su dosta dobro klasifikovani - broj tačno klasifikovanih je dosta veliki, za razliku od onih koji to nisu.

3.2 Klasifikacija korišćenjem algoritma Random forest

```
Velicina skupa za obucavanje: 225
Velicina skupa za testiranje: 99
```

```
# eksperimentalno je utvrdjeno da algoritam
# daje najbolje rezultate kada se poziva
# sa parametrom 31

clf = RandomForestClassifier(n_estimators=31)

# 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))
```

Preciznost je jako loša kada se koristi ovaj algoritam.

```
train preciznost: 0.6266666666666667
test preciznost: 0.36363636363636365
```

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       0.50      0.33      0.40         6
          1       0.31      0.56      0.40         9
          2       0.36      0.44      0.40         9
          3       0.50      0.11      0.18         9

avg / total       0.41      0.36      0.34        33

Train izvestaj:
             precision    recall  f1-score   support

          0       0.50      0.33      0.40         9
          1       0.69      0.73      0.71        30
          2       0.57      0.81      0.67        21
          3       0.71      0.33      0.45        15

avg / total       0.64      0.63      0.61        75

Matrica konfuzije za skup za obucavanje:
[[ 3  4  2  0]
 [ 2 22  5  1]
 [ 1  2 17  1]
 [ 0  4  6  5]]
```

```
Matrica konfuzije za skup za testiranje:
[[2 4 0 0]
 [2 5 2 0]
 [0 4 4 1]
 [0 3 5 1]]
```

3.3 Klasifikacija korišćenjem stabla odlučivanja (kriterijum: entropija)

```
Velicina skupa za obucavanje: 225
Velicina skupa za testiranje: 99
```

```
# vrsi se klasifikacija koriscenjem entropije
clf = DecisionTreeClassifier(criterion='entropy')

# 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))
```

Preciznost je jako loša kada se koristi i ovaj algoritam.

```
train preciznost: 0.6
test preciznost: 0.30303030303030304
```

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       0.25      0.33      0.29         3
          1       0.47      0.50      0.48        14
          2       0.25      0.09      0.13        11
          3       0.10      0.20      0.13         5

avg / total       0.32      0.30      0.30        33

Train izvestaj:
             precision    recall  f1-score   support

          0       0.57      0.33      0.42        12
          1       0.55      0.92      0.69        25
          2       0.75      0.32      0.44        19
          3       0.67      0.63      0.65        19

avg / total       0.63      0.60      0.57        75

Matrica konfuzije za skup za obucavanje:
[[ 4  8  0  0]
 [ 1 23  0  1]
 [ 2  6  6  5]
 [ 0  5  2 12]]

Matrica konfuzije za skup za testiranje:
[[1 1 1 0]
 [1 7 0 6]
 [2 5 1 3]
 [0 2 2 1]]
```

3.4 Klasifikacija korišćenjem stabla odlučivanja (kriterijum: Ginijev indeks)

```
Velicina skupa za obucavanje: 225
Velicina skupa za testiranje: 99
```

```
# vrsi se klasifikacija koriscenjem Ginijevog indeksa
clf = DecisionTreeClassifier(criterion='gini')

# 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))
```

Preciznost je jako loša kada se koristi i ovaj algoritam, približna onoj kada je parametar koji se zadaje algoritmu za stabla odlučivanja entropija.

```
train preciznost: 0.6133333333333333
test preciznost: 0.3333333333333333
```

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       0.14      0.50      0.22         2
          1       0.47      0.58      0.52        12
          2       0.38      0.23      0.29        13
          3       0.00      0.00      0.00         6

avg / total       0.33      0.33      0.31        33

Train izvestaj:
             precision    recall  f1-score   support

          0       0.50      0.62      0.55        13
          1       0.62      0.78      0.69        27
          2       0.62      0.47      0.53        17
          3       0.75      0.50      0.60        18

avg / total       0.63      0.61      0.61        75

Matrica konfuzije za skup za obucavanje:
[[ 8  4  1  0]
 [ 4 21  1  1]
 [ 3  4  8  2]
```

```
 [ 1  5  3  9]]

Matrica konfuzije za skup za testiranje:
[[1 1 0 0]
 [3 7 1 1]
 [3 5 3 2]
 [0 2 4 0]]
```

4 Klasterovanje

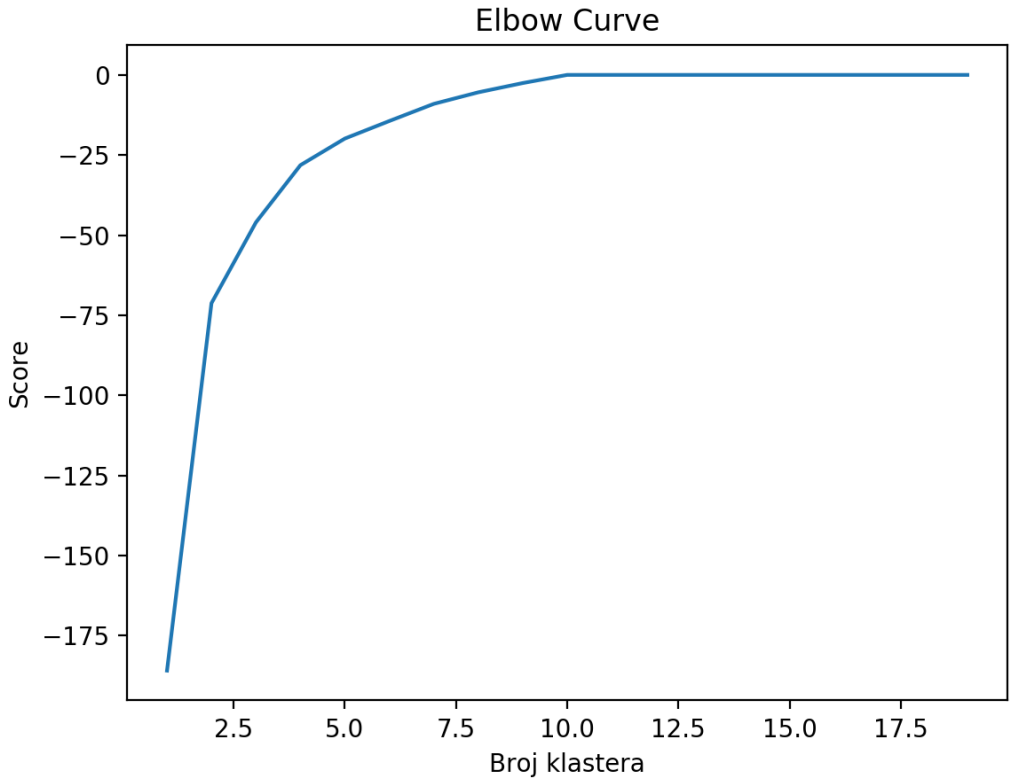
4.1 Algoritam K-sredina

Probaćemo da klasterujemo podatke na osnovu nekoliko relevantnih atributa.

```
# podaci se ucitavaju
df = pd.read_csv('./food-choices/food_coded.csv')

# podaci se sredjuju za koriscenje
data = data[data['weight'].apply(lambda x: str(x).isdigit())]
Y = data[['Gender', 'eating_out']]
X = data[['weight']]
```

Za početak, treba nam broj klastera. Postoji kriva, takozvana elbow curve koja se koristi kao pomoć pri određivanju broja klastera.
Traži se mesto na kom se kriva drastično prelama i ne napreduje. To nam govori da veći broj klastera od broja na ne bi ništa značajno doneo, ne bi ništa bolje objasnio, rasporedio podatke.

```
# pravi se niz od 20 brojeva
# 20 puta se primeni algoritam K-sredina
# za razlicit broj klastera - od 0 do 20

n = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in n]

score = [kmeans[i].fit(Y).score(Y) for i in range(len(kmeans))]

# dobijeni rezultati se iscrtavaju

plt.plot(n, score)
plt.xlabel('Broj klastera')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



Sa slike se ne može jasno videti koja je tačka ona koja nam tačno može odrediti broj klastera. Ovaj slučaj se u praksi često dešava i tada se koriste Bartletov (eng. Bartlette) i Levenijev (eng. Levene) test.
Kako se dvoumimo oko 2 tačke, pokrenućemo algoritam za obe i videti koja klaster analiza izgleda bolje.

```
# PCA algoritam se koristi da bi
# se podaci koji su mozda
# previse rasprseni
# pretvorili u linearne kombinacije
# i na taj nacin bili lakse citljivi

pca = PCA(n_components=2).fit(Y)
pca_d = pca.transform(Y)
pca_c = pca.transform(X)

kmeans = KMeans(n_clusters=8)

# za rasporedjivanje podataka
# u 3 klastera bi se prethodna linija
# zamenila narednom zakomentarisanom linijom
# kmeans = KMeans(n_clusters=3)
```

```
kmeansoutput = kmeans.fit(Y)

# mogu se pogledati vrednosti
# krajnjih centroida

print(kmeans.cluster_centers_)
```

Dobijeni centroidi za 8 klastera su:

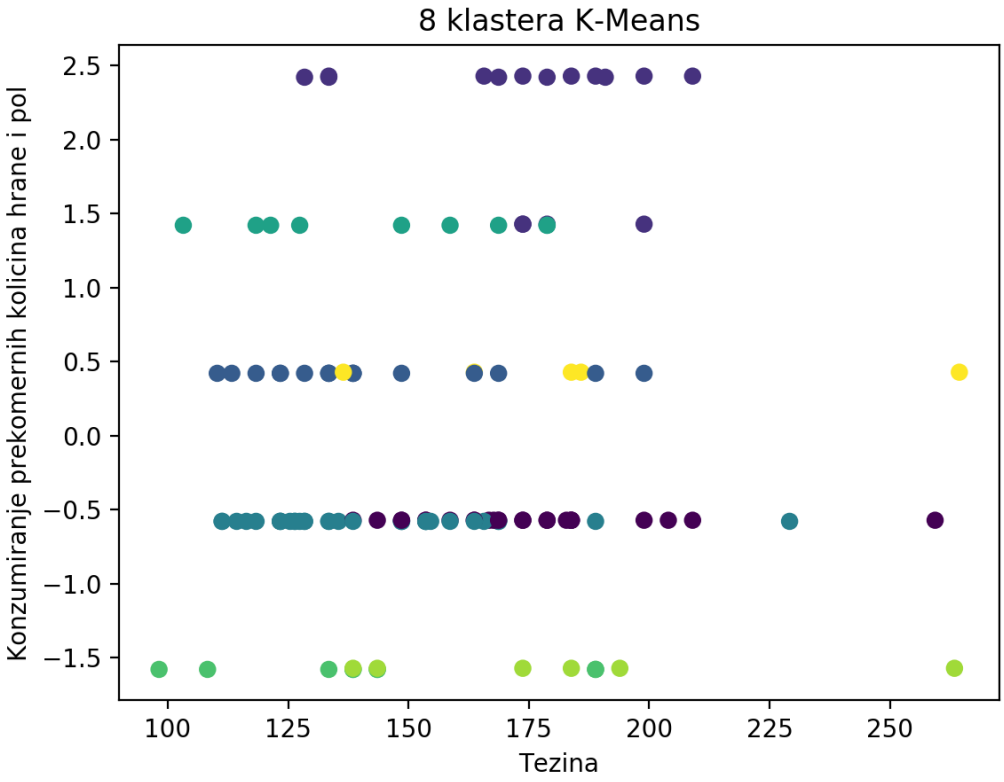
```
[[2.     2.    ]
 [1.6875 4.75  ]
 [1.     3.    ]
 [1.     2.    ]
 [1.     4.    ]
 [1.     1.    ]
 [2.     1.    ]
 [2.     3.    ]]
```

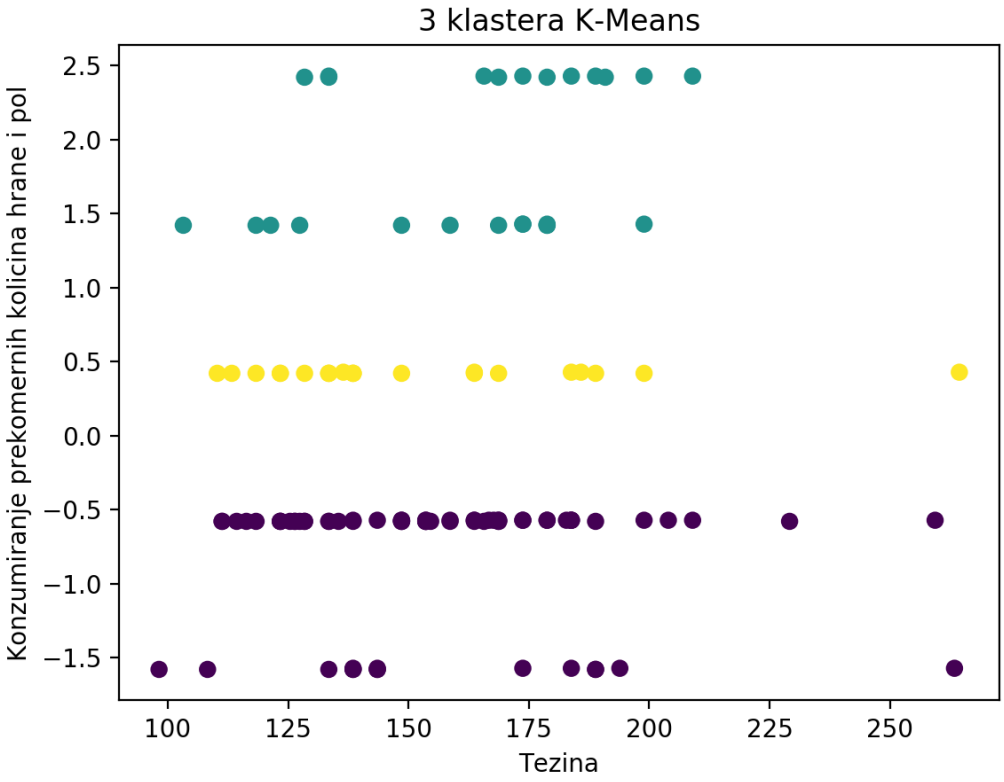
Dobijeni centroidi za 3 klastera su:

```
[[1.775      1.625     ]
 [1.09090909 2.4       ]
 [1.44       4.48      ]]
```

```
# klasteri se prikazuju

plt.figure('8 klastera K-Means')
plt.scatter(pca_c[:, 0], pca_d[:, 0], c=kmeansoutput.labels_)
plt.xlabel('Tezina')
plt.ylabel('Konzumiranje prekomernih kolicina hrane i pol')
plt.title('8 klastera K-Means')
plt.show()

plt.figure('3 klastera K-Means')
plt.scatter(pca_c[:, 0], pca_d[:, 0], c=kmeansoutput.labels_)
plt.xlabel('Tezina')
plt.ylabel('Konzumiranje prekomernih kolicina hrane i pol')
plt.title('3 klastera K-Means')
plt.show()
```

Dobijeni klasteri kada sa algoritam K-sredina pokrene za 8 i za 3 klastera:


8 klastera K-Means
Konzumiranje prekomernih kolicina hrane i pol
Tezina
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
100
125
150
175
200
225
250


3 klastera K-Means
Konzumiranje prekomernih kolicina hrane i pol
Tezina
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
100
125
150
175
200
225
250

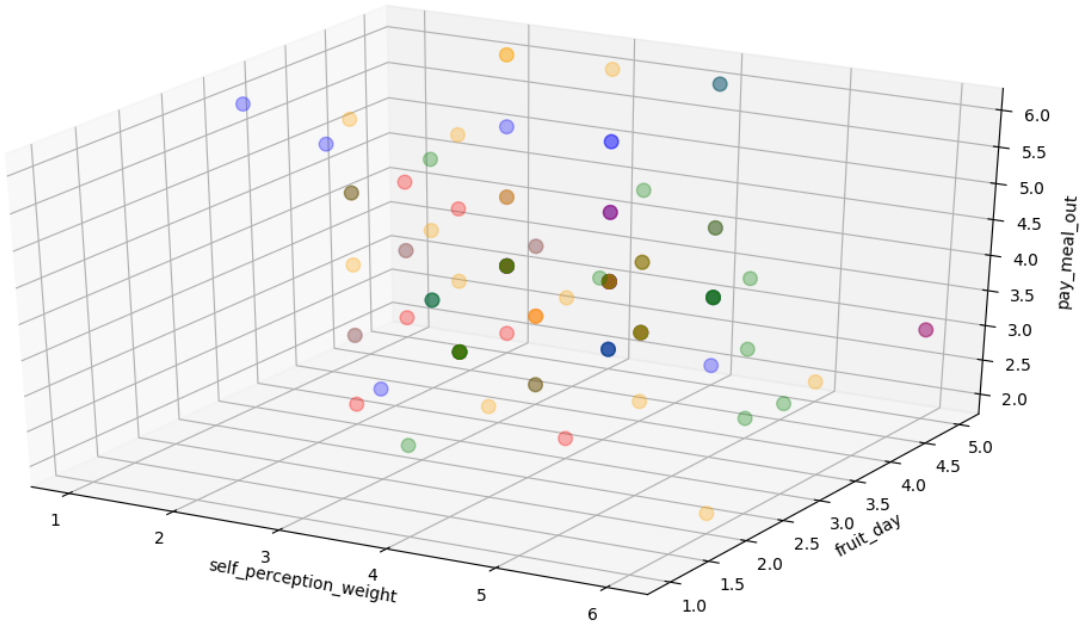
5 Vizuelizacija podataka

Atributi koje posmatramo:

- self_perception_weight
  Procena sopstvene težine
- fruit_day
  Koliko je verovatno da bi pojeli voće u toku dana
- pay_meal_out
  Koliko bi platili za jedan obrok?
- weight
  Težina

```
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()
```



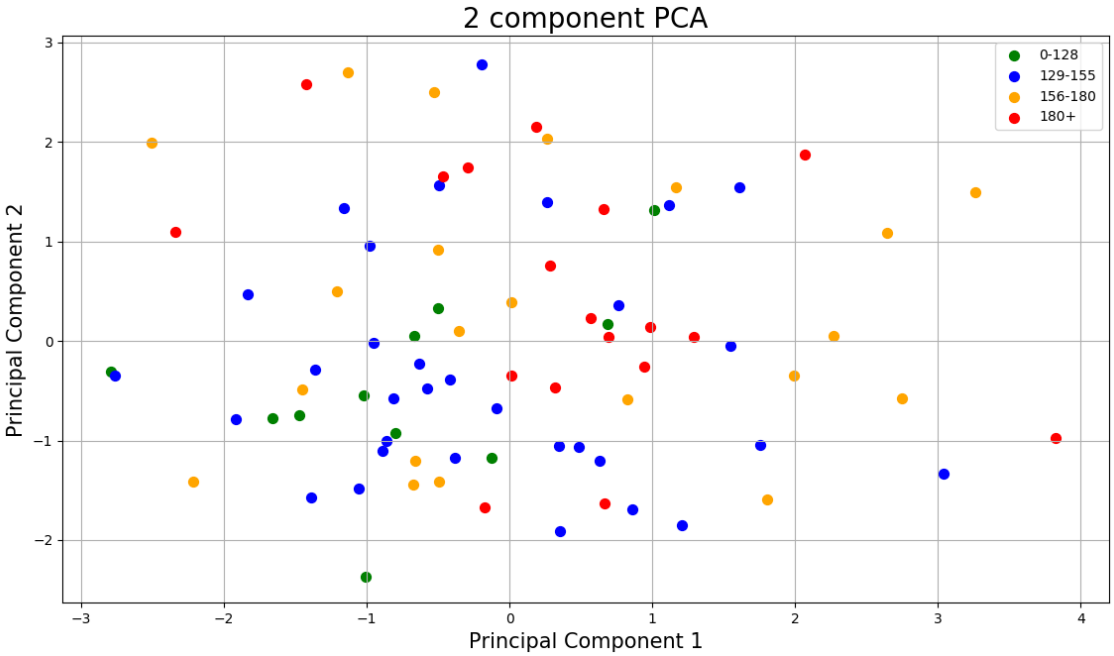
5.1 PCA

Korišćenjem PCA algoritma ćemo redukovati skup na manji, zadržavajući većinu informacija iz inicijalnog skupa podataka.

```
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

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

# odabir atributa

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]]

# obrada podataka
# brisanje redova sa nedostajucim vrednostima,
# konvertovanje podataka u odgovarajuce
# kako bi se moglo raditi sa njima

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

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)

# podaci su tipa float, a potrebno je
# da budu tipa int, pa se u narednim
# koracima vrsi konverzija

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)

# transformisanje ciljnog atributa
# preslikacemo sve vrednosti weight atributa
# u skup {0, 1, 2, 3}
# ispitanici:
# sa manje od 58.5kg spadaju u grupu 0
# izmedju 58.5 i 70kg spadaju u grupu 1
# izmedju 70 i 81.6kg spadaju u grupu 2
# sa preko 81.6kg spadaju u grupu 3

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()
```

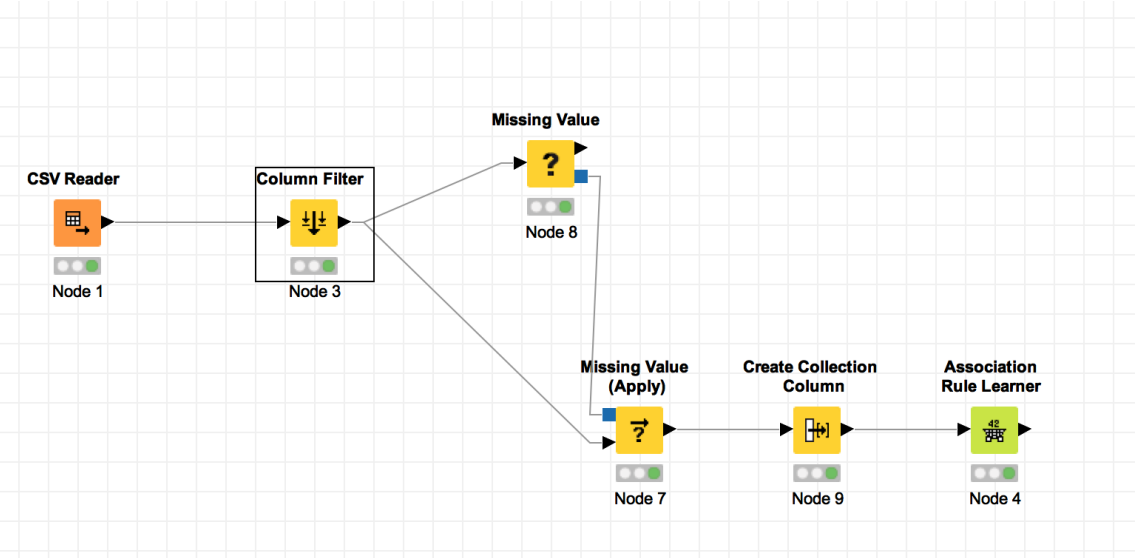


Na osnovu grafika zaključujemo da najveći broj ispitanika ima između 58.5kg i 70kg.

6 Pravila pridruživanja

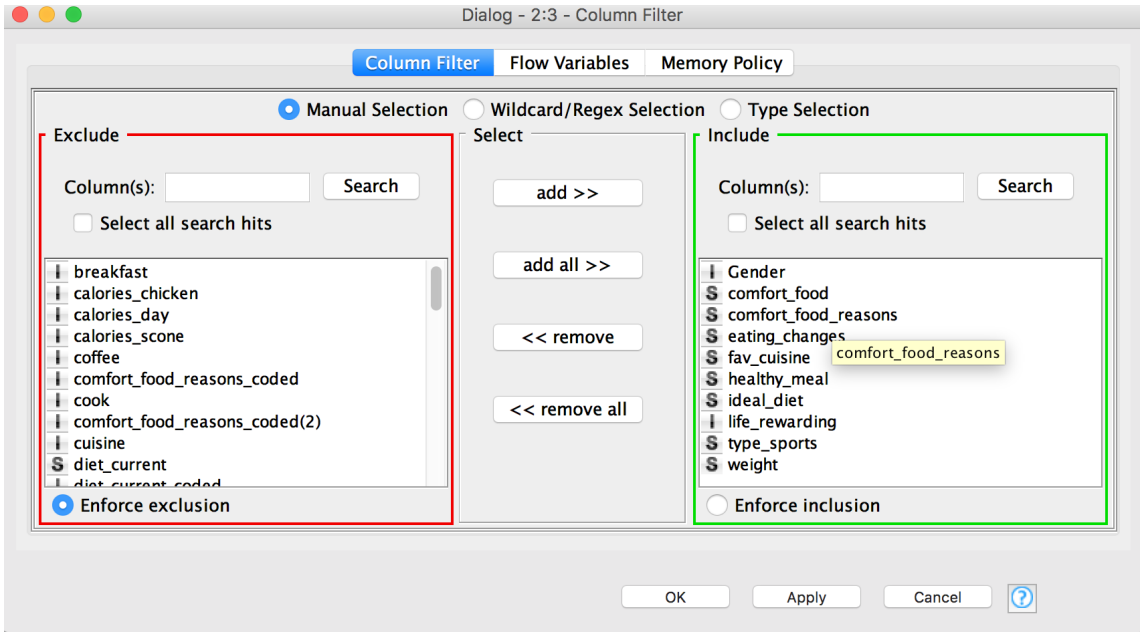
Za pravila pridruživanja je korišćen alat Knime.

Koristeći čvorove CSV Reader za učitavanje skupa podataka, Column Filter za odabir atributa nad kojima želimo da primenimo pravila pridruživanja, Missing Value i Missing Value Apply kako bismo se pobrinuli o nedostajućim vrednostima, Create Collection Column za pravljenje liste od odabranih atributa i Association Rule Learner kako bismo pronašli česte skupove, pokušaćemo da pronađemo neke česte skupove.



Kako u skupu postoji veliki broj nedostajućih vrednosti, birajući da ih uklonimo bismo izgubili veliki broj podataka koji nam u ovom slučaju mogao biti od koristi. Zato su za sve nedostajuće vrednosti zamenjene najfrekventnijim vrednostima iz odgovarajućih kolona.

Atributi za koje je smatrano da bi mogli biti u korelaciji su pol, težina, sport kojim se bave, hrana koju vole da jedu, iz kog razloga jedu hranu za utehu, omiljena kuhinja, koliko je za njih život ispunjavajuć, kako im se promenio način ishrane od kad su na koledžu i šta za njih predstavlja zdrav obrok.
Odabir atributa:



Dobijena pravila priduživanja:

Frequent itemsets/Association rules - 2:4 - Association Rule Learner

File Hilite Navigation View

Table "default" - Rows: 28 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	Support	Confid...	Lift	Conse...	implies	Items
rule4	0.064	1	1.471	1	<---	[135]
rule2	0.056	0.467	1.357	Hockey	<---	[3]
rule3	0.056	0.163	1.357	3	<---	[Hockey]
rule7	0.064	0.421	1.224	Hockey	<---	[1,2]
rule18	0.12	0.789	1.161	1	<---	[Italian]
rule19	0.12	0.176	1.161	Italian	<---	[1]
rule20	0.144	0.75	1.103	1	<---	[Italian]
rule21	0.144	0.212	1.103	Italian	<---	[1]
rule10	0.072	0.209	1.09	Italian	<---	[Hockey]
rule11	0.072	0.375	1.09	Hockey	<---	[Italian]
rule14	0.088	0.733	1.078	1	<---	[3]
rule15	0.088	0.129	1.078	3	<---	[1]
rule16	0.096	0.5	1.059	2	<---	[Italian]
rule17	0.096	0.203	1.059	Italian	<---	[2]
rule24	0.168	0.488	1.035	2	<---	[Hockey]
rule25	0.168	0.356	1.035	Hockey	<---	[2]
rule26	0.24	0.698	1.026	1	<---	[Hockey]
rule27	0.24	0.353	1.026	Hockey	<---	[1]
rule8	0.072	0.643	0.945	1	<---	[7]
rule9	0.072	0.106	0.945	7	<---	[1]
rule12	0.088	0.611	0.899	1	<---	[8]
rule13	0.088	0.129	0.899	8	<---	[1]
rule0	0.056	0.389	0.824	2	<---	[8]
rule1	0.056	0.119	0.824	8	<---	[2]
rule5	0.064	0.267	0.565	2	<---	[1,Hockey]
rule6	0.064	0.381	0.56	1	<---	[2,Hockey]
rule23	0.152	0.322	0.474	1	<---	[2]
rule22	0.152	0.224	0.474	2	<---	[1]

Kao što se može videti, nijedno pravilo nije od velikog značaja jer lift mera nijednog pravila ne odstupa od jedinice za neku značajnu vrednost. Ipak, nešto se može zaključiti iz pravila na osnovu pojavljivanja određenih vrednosti: kao i do sada, italijanska hrana je ubedljivo najčešći odgovor kada je se radi o omiljenoj hrani ispitanika, a sport koji se najčešće pojavljuje kao onaj kojim se bave je hokej.