Accidental Deaths and Injuries Data Analasys

Anja Miletić 97/2015

Mateja Marjanović 172/2015

August 27, 2018

Contents

1 Uvod

Podaci su skinuti na linku http://www.gunviolencearchive.org/reports. Podaci se nalaze u dve datoteke: deaths.csv i injuries.csv i predstavljaju izvestaje o slucajnim povredama i smrtnim slucajevima nastalim koriscenjem vatrenog oruzja u Sjedinjenim Americkim Drzavama.

2 Analiza i pretprocesiranje podataka

Obe datoteke sadrze tabele sa istim kolonama. Opisi kolona su dati u tabeli 1.

Incident Date	datum incidenta u obliku Mesec dan, godina
State	savezna americka drzava u kojoj se dogodio incident
City or County	grad ili okrug u kome se dogodio incident
Adress	adresa na kojoj se dogodio incident
# Killed	broj ubijenih osoba
# Injured	broj povredjenih osoba

2.1 Analiza podataka

Pogledajmo detaljnije nase dve datoteke. Koristimo Python kod da izlistamo osnovne informacije o podacima.

```
import pandas

print("**************\n*deaths.csv*\n**************\n")
df_deaths = pandas.read_csv("deaths.csv")
print(df_deaths.head())
print(df_deaths.count())
print(df_deaths.describe())
for col in df_deaths.columns:
    print("count values in column " + col)
    print(df_deaths[col].value_counts(dropna = False))
```

rezultat izvrsavanja:

```
**************
*deaths.csv*
**************

     Incident Date  State  City Or County  Address # Killed # Injured Operations
0  August 17, 2018 Louisiana West Monroe Philpot Rd  1      0     NaN
1  August 16, 2018 Virginia Richmond 5600 .. Ct      1      0     NaN
2  August 15, 2018 Kentucky  Louisville   1708 .. St  1      0     NaN
3  August 15, 2018 Ohio      Columbus 1200 .. Rd     1      0     NaN
4  August 14, 2018 Michigan Pontiac 40 .. Salee Ln   1      0     NaN
```

```
Incident Date   500
State           500
City Or County  500
Address         468
# Killed        500
# Injured       500
Operations        0

       # Killed  # Injured Operations
count  500.00000 500.000000       0.0
mean     1.00600   0.038000       NaN
std      0.09992   0.211294       NaN
min      0.00000   0.000000       NaN
25%      1.00000   0.000000       NaN
50%      1.00000   0.000000       NaN
75%      1.00000   0.000000       NaN
max      2.00000   2.000000       NaN

count values in column Incident Date
March 4, 2018       6
July 1, 2017        5
March 12, 2018      5
October 29, 2017    5
November 2, 2017    4
September 30, 2017  4
June 24, 2017       4
May 22, 2018        4
January 1, 2018     4
March 31, 2018      4
February 20, 2018   4
July 4, 2017        4
October 7, 2017     4
June 1, 2017        4
June 8, 2017        4
..
count values in column State
Texas           48
Tennessee       30
Florida         26
Georgia         24
Missouri        23
Ohio            22
Alabama         22
Louisiana       21
Mississippi     21
South Carolina  19
Pennsylvania    17
..
count values in column # Killed
```

```
1    495
2      4
0      1
count values in column # Injured
0    483
1     15
2      2
count values in column Operations
NaN   500
```

Iz ovih rezultata mozemo primetiti vise stvari. Tabela ima tacno 500 unosa, pri cemu kolone Incident Date, State, City or County, #Killed i #Injured nemaju nijednu null vrednost. Kolona Operations nema nijednu vazecu vrednost i ona ce biti izbacena iz daljeg razmatranja. Najvise unosa je bilo 4. marta 2018, a najvise incidenata se dogodilo u Teksasu, cak 48. Kolonu koja predstavlja adrese cemo u nastavku ignorisati.

```
#remove column Adress and Operations
df = df[['Incident Date','State','City Or County','# Killed','# Injured']]
```

Primetimo da u datoteci deaths.csv postoje unosi gde je bilo 0 smrtnih slucajeva. Analogno, u datoteci injuries.csv postoje unosi u kojima nije bilo povredjenih. Ove redove tabele zelimo da izbacimo. Ovo postizemo uz pomoc sledece dve funckije:

```
import pandas

def no_deaths():
    print('put Null values on zero death rows')
    df_deaths = pandas.read_csv('./deaths.csv')
    for i, row in df_deaths.iterrows():
        number_deaths = row['# Killed']
        if(number_deaths == 0):
            df_deaths.set_value(i, '# Killed', None)

    return df_deaths

def remove_missing_values(df):
    df_deaths_tmp = df[pandas.notnull(df['# Killed'])]
    return df_deaths_tmp
```

2.2 Distribucija frekvencija podataka

U ovom delu cemo se fokusirati na prvu kolonu tabele, Incident Date. Zelimo da analiziramo frekvenciju incidenata u odnosu na dan u nedelji, mesec i slicno. Da bi mogli da koristimo unose iz ove kolone moramo da transformisemo datum iz jedne niske u brojcane vrednosti za dan, mesec i godinu koje onda mozemo da cuvamo u nizovima. Sledeca funkcija konvertuje datum u datetime format:

```
import pandas
from datetime import datetime as dt

def dayofweek(df):
   for i, row in df.iterrows():
      date = row["Incident Date"]
      year_sep = date.split(',')
      date_sep = year_sep[0].split(' ')

      month = months[date_sep[0].strip()]
      day = int(date_sep[1].strip())
      year = int(year_sep[1].strip())

      readable_date = dt.date(year, month, day)

      # 0 - Monday, 6 - Sunday
      y[readable_date.weekday()] += 1
      y_month[month-1] += 1
      if month == 7:
         dayOfJuly[day-1] += 1
```

Funckija dayofweek istovremeno kategorise unose prema danu u nedelji i mesecu. Niz dayOfJuly sadrzi sve incidente koji su se dogodili u julu kategorisani po danu. Koristimo Python biblioteku matplotlib da graficki prikazemo dobijene rezultate.

```
df = pd.read_csv("processed_deaths.csv")

   dayofweek(df)
   print("number of incidents each day of week:")
   print(y)
   plt.bar(x, y, width=0.5, color="blue", tick_label=labels)
   plt.show()

   print("number of incidents per month:")
   print(y_month)
   plt.bar(x_month, y_month, width=0.5, color="red", tick_label=labels_month)
   plt.show()

   print("number of incidents per day in july:")
   print(dayOfJuly)
   plt.bar(range(31), dayOfJuly, width=0.5, color="green",
       tick_label=labels_july)
   plt.show()
```

Izlaz u konzoli:

```
number of incidents each day of week:
[ 61.  61.  68.  64.  62. 102.  81.]
number of incidents per month:
```

```
[29. 24. 33. 17. 49. 77. 81. 49. 32. 45. 31. 32.]
number of incidents per day in july:
[5. 2. 1. 6. 7. 2. 3. 4. 1. 1. 0. 3. 3. 4. 2.
0. 1. 1. 2. 3. 1. 4. 2. 4. 0. 3. 3. 2. 4. 4. 3.]
```

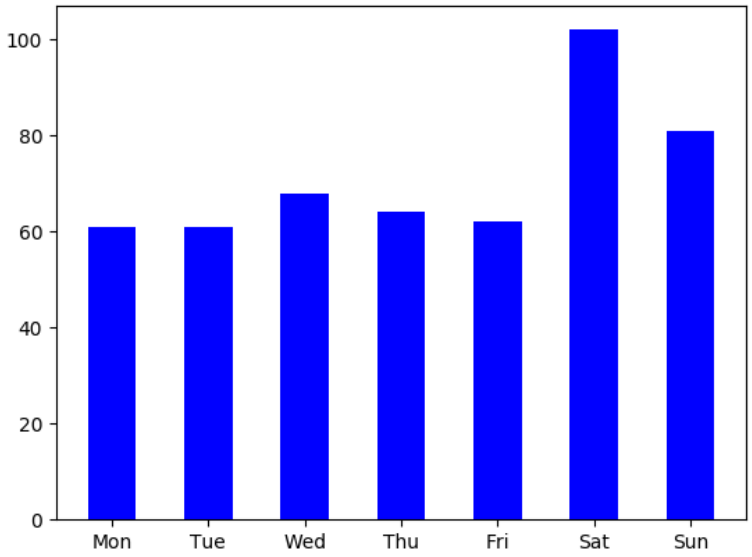

Figure 1: Distribucija po danu u nedelji

Dolazimo do nekoliko zakljucaka; najpre primecujemo ubedljivo najvecu frekvenciju podataka subotom, zatim nedeljom. Ovo zapazanje ima smisla jer ljudi vikendom uglavnom ne rade pa stoji da ce se najvise nesreca dogoditi upravo tad. Zatim, figura 2 pokazuje jun i jul kao mesece u kojima je zabelezeno najvise incidenata, znatno vise nego ostalih meseci. Izdvojili smo jul kao mesec od posebnog interesovanja (figura 3). Najvise unosa postoji 4. i 5. jula, iako su ovo bili radni dani. Zasto? Upravo tih dana je drzavni praznik u SAD, proslava dana nezavisnosti. Tradicionalno postoji mnogo veca upotreba oruzja u svrhu proslave ovih dana, a samim tim i veci broj zabelezenih povreda i slucajnih smrtnih ishoda. Vredi napomenuti da je 2. novembra, na Dan mrtvih koji obelezavaju pripadnici latino populacije takodje zabelezen veci broj nesreca od ocekivanog.

2.3 Legislacije o vatrenom oruzju u SAD

Savezne drzave mogu da definisu svoje zakone koji regulisu upotrebu oruzja. Postoji velika raznolikost u broju ovih zakona (legislacija) medju drzavama. Zanimalo nas je da li postoji veza izmedju broja zakona i broja incidenata, tj. da li veci broj zakona znaci veci broj slucaja i obratno. U te svrhe smo napravili novu .csv datoteku:

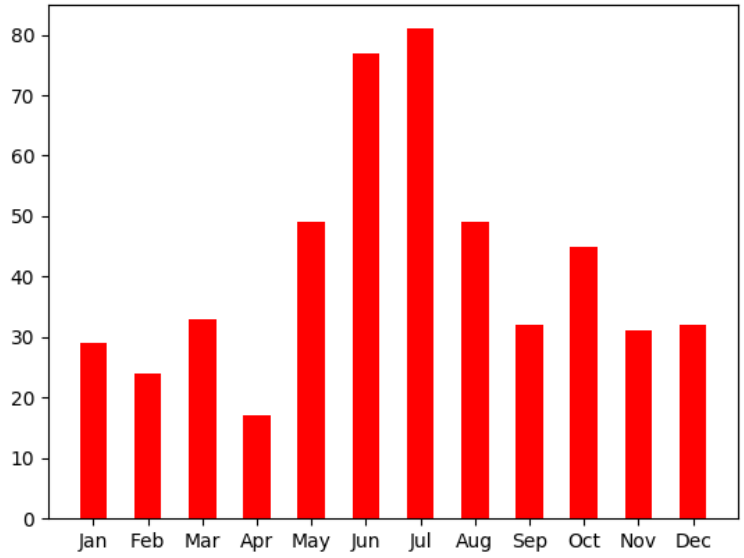

Figure 2: Distribucija po mesecima

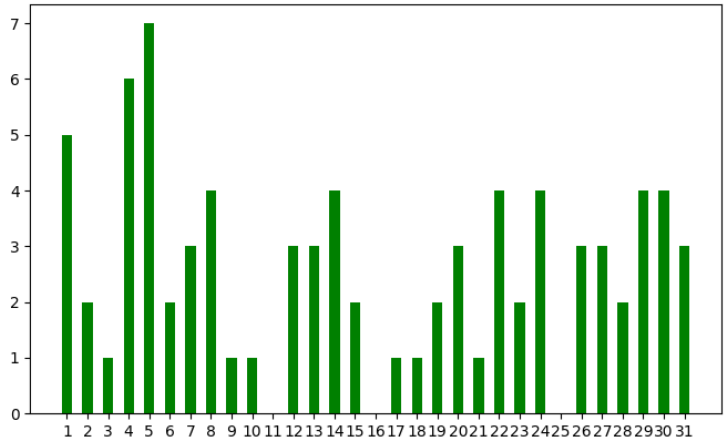

Figure 3: Distribucija tokom jula

Listing 1: gun_laws_per_state.csv

```
"State","# Laws"
"Alabama",10
"Alaska",3
"Arizona",8
"Arkansas",11
"California",107
"Colorado",30
"Connecticut",90
"Delaware",40
"Florida",21
...
```

Podaci su dobijeni sa sajta Americke vlade https://www.statefirearmlaws.org/national-data. Sada mozemo da sumiramo podatke i generisemo grafikon(figura 4):

Listing 2: processing_gun_laws.py

```
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
import numpy as np

# number of gun laws
x = np.zeros(50)
# number of deaths
y = np.zeros(50)

def main():
   df_laws = pd.read_csv("gun_laws_per_state.csv")
   df_deaths = pd.read_csv("processed_deaths.csv")

   for i, row in df_laws.iterrows():
      subsamples = df_deaths.loc[df_deaths['State'] == row['State']]
      total_deaths = np.sum(subsamples['# Killed'])

      x[i] = df_laws.get_value(i, '# Laws')
      y[i] = total_deaths

   plt.scatter(x, y, color="blue")
   plt.show()

if __name__ == "__main__":
   main()
```

Posmatrajuci grafikon primecujemo grupisanje podataka u donjem levom kvadrantu. Dakle, mali broj zakona i mali broj slucajeva. Medjutim, autori smatraju da se na osnovu ovih podataka ne mogu doneti cvrsti zaključci o vezi izmedju broja legislacija i broja nesrecnih slucajeva.

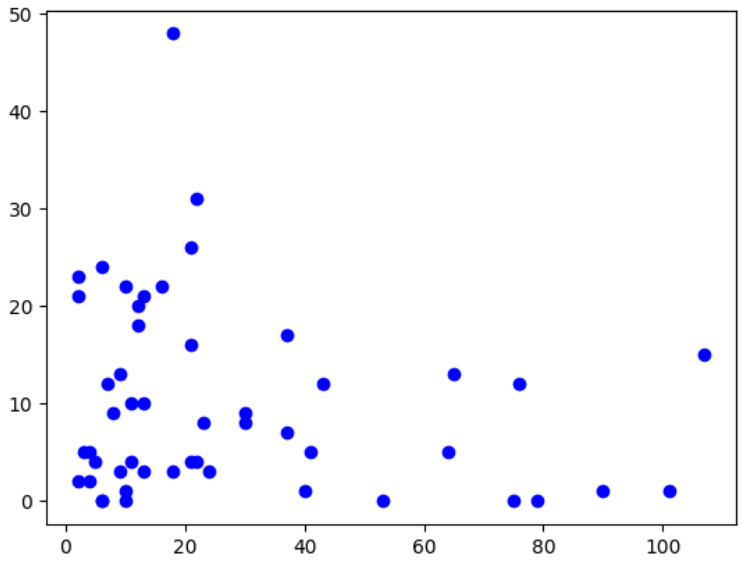

Figure 4: odnos zakona i zabelezenih slucajeva

3 Klasterovanje

Proces klasterovanja izvrsen je na podacima koji su pre samog izvrsavanja pripremljeni (pretprocesiranje i namestanje podataka). Podaci koji su prosledjeni imaju dva atributa, to su 'State' i 'Number of Incidents', 'State' predstavlja drzavu, a 'Number of Incidents' predstavlja ukupan broj incidenata (slucajnih povreda i slucajnih ubistava) u toj drzavi.

3.1 Pripremanje podataka za klasterovanje

Da bismo izvrsili klasterovanje koje ima smisla, gde moze da se vidi koji podaci su slicniji, a koji razlicitiji, moramo da prvo te podatke izmenimo na neki nacin i namestimo ih da budu takvi da nam odgovaraju. U narednom kodu napisanom u Python-u smo izmenili podatke iz processed_deaths.csv i processed_injuries.csv i napravili novi fajl summedIncidentsPerState.csv.

```
def changeDateFormat(df):
   for i, row in df.iterrows():
      month = row["Incident Date"].strip().split(' ')[0].split(',')[0]
      df.set_value(i, "Incident Date", month)

   df = df.drop(columns = ["Unnamed: 0", "Incident Date", "City Or County"])
```

```
allIncidents = {}
allStates = []
for i, row in df.iterrows():
   state = row["State"].strip()
   if state not in allStates:
      allStates.append(state)
      allIncidents[state] = int(row["# Injured"]) + int(row["# Killed"])
   else:
      allIncidents[state] += int(row["# Injured"]) + int(row["# Killed"])
   df = df.drop([i])

df = df.drop(columns = ["# Killed", "# Injured"])
for k, v in allIncidents.iteritems():
   state = k
   numberOfDeaths = v
   df = df.append({"State" : state, "# Incidents" : int(numberOfDeaths)},
       ignore_index = True)

return df
```

3.2 Prosledjivanje generisanih podataka i klasterovanje

Sad kad smo generisali podatke koristimo ih u Knime-u. Prvo citamo podatke sa CSV Reader-om, zatim ih saljemo na normalizaciju, tj. da se vrednosti iz kolone sa brojem ubistava skaliraju na interval [0, 1]. Posle toga racunamo Euklidska rastojanja podataka, pa ta rastojanja i izlaz iz cvora koji je normalizovao podatke saljemo DBSCAN cvoru koji dodaje jos jednu kolonu i svakoj instanci dodeljuje neku vrednost u zavisnosti od toga gde se nalazi. Ostalo je samo jos da se iscrtaju podaci, da bismo to uradili koristimo Color Manager i Scatter Plot.

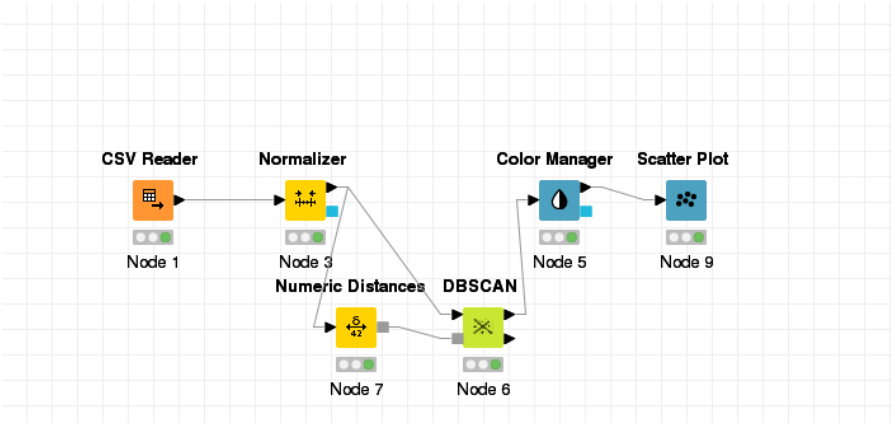

Figure 5: Izgled klasterovanja podataka u Knime-u

3.3 Izgled klasterovanih podataka i analiza

Sada imamo podatke koji su podeljeni, svaki pripada nekom klasteru (neki pripadaju klasteru Noise) i svakom klasteru je dodeljena druga boja. Moze se primetiti da je drzava u kojima ima vise ubistava mnogo manje nego drzava u kojima ima imanje. Npr. u Teksasu ima toliko vise nego bilo gde drugde da se Teksas dozivljava kao sum.

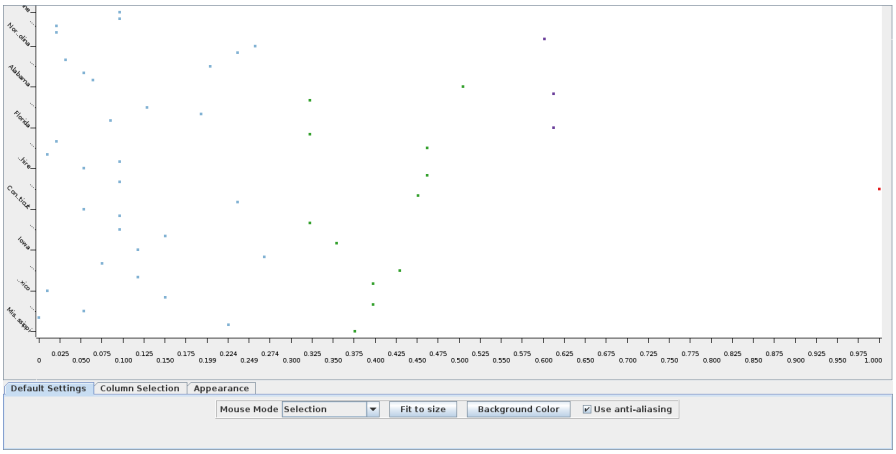

Figure 6: Klasterovani podaci koji prikazuju zavisnost drzave i broja ubistava u toj drzavi

4 Klasifikacija

Proces klasifikacije ili nadgledanog ucenja slicno kao i klasterovanje izvrsavamo na podacima koje smo generisali u drugom programu, zatim smo u Knime-u te podatke iskoristili i izvrsili klasifikaciju. Uradjene su dve klasifikacije. Jedna je predvidjanje da li ce u odredjenom danu u nedelji u odredjenom mesecu biti malo, srednje ili puno incidenata. Malo incidenata znaci da se u tom mesecu, u tom danu u nedelji (npr. svakom ponedeljku) ukupno desilo manje od 8, srednje znaci da je taj broj izmedju 8 i 15, a puno vise od 15. Sa ovim brojevima ima slican broj parova (dan u nedelji, mesec) (blizu 33% svako). To nije slucajnost, s obzirom na to da je uzoracka sredina broja ubistava 12.8. U drugoj klasifikaciji izvrsava se predvidjanje da li ce biti puno (vise od 1) ili malo (manje jednako 1) incidenata u nekoj drzavi u nekom gradu.

4.1 Klasifikacija: Dan u nedelji, mesec, broj incidenata

Kao i ranije prvo podatke zelimo da transformisemo i dobijemo zeljeni oblik. U ovom slucaju zeljeni oblik bi bio da imamo kolonu mesec, dan u nedelji i ukupan broj incidenata u toj kombinaciji (meseca i dana u nedelji). Sledeci kod ce generisati novi fajl u kome ce se nalaziti podaci.

```
def changeDateFormat(df):
   df = df.drop(columns = ["Unnamed: 0", "Incident Date"])

   incidents = {}
   for i, row in df.iterrows():
      state = row["State"].strip()
      city = row["City Or County"].strip()
      if state + ":" + city not in incidents:
         incidents[state + ":" + city] = int(row["# Injured"]) + int(row["#
             Killed"])
      else:
         incidents[state + ":" + city] += int(row["# Injured"]) + int(row["#
             Killed"])
      df = df.drop([i])

   df = df.drop(columns = ["# Killed", "# Injured"])
   for k, v in incidents.iteritems():
      key = k.split(':')
      city = key[1]
      state = key[0]
      numberOfDeaths = v
      df = df.append({"State" : state, "City Or County" : city, "# Incidents" :
          int(numberOfDeaths)}, ignore_index = True)

   return df
```

4.1.1 Klasifikacija u Knime-u

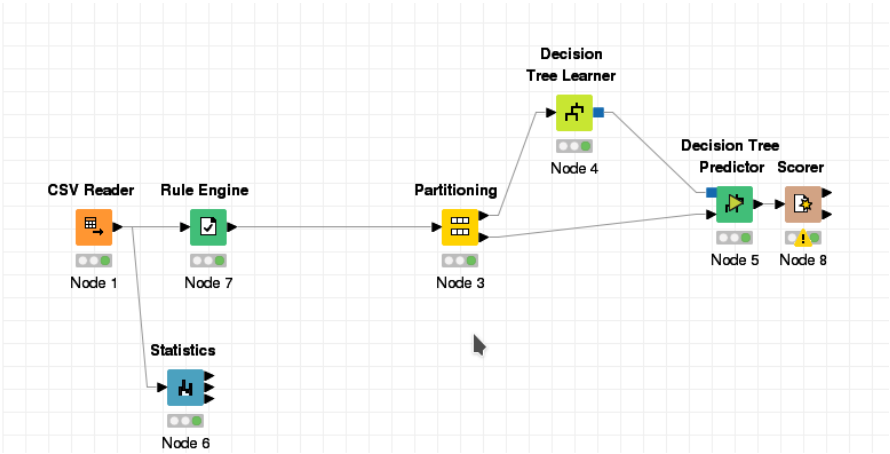

Figure 7: Izlged klasifikacije u Knime-u

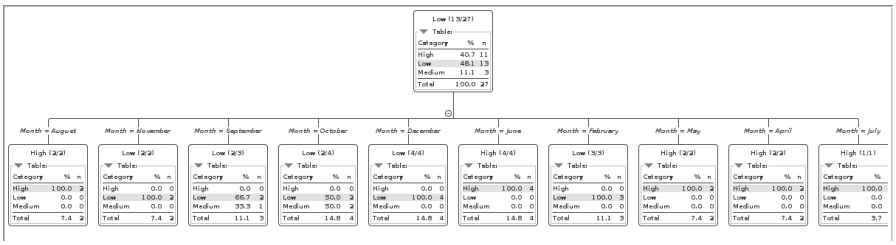

Figure 8: Drvo odlucivanja u kome se na osnovu dana u nedelji i meseca predvidja da li ce biti malo, srednje ili puno incidenata

Predictio...	High	Low	Medium
High	18	0	6
Low	0	19	0
Medium	0	0	0

Correct classified: 37 Wrong classified: 6

Accuracy: 86.047 % Error: 13.953 %

Cohen's kappa (κ) 0.756

Figure 9: Matrica konfuzije

4.2 Klasifikacija: Drzava, grad ili mesto, broj incidenata

Slicno kao i u prethodnoj klasifikaciji uzimamo procesuirane podatke, generisemo nove na osnovu njih, zatim saljemo u Knime da izvrsi klasifikaciju. Glavna razlika je u tome sto se ovde radi sa drzavama, gradovima i brojem incidenata. Cilj nam je da predvidimo da li ce biti puno (vise od jednog incidenta) ili malo (samo jedan incident).

4.2.1 Generisanje novih podataka

```
def sumIncidentsPerCityAndState(df):
df = df.drop(columns = ["Unnamed: 0", "Incident Date"])

incidents = {}
for i, row in df.iterrows():
   state = row["State"].strip()
   city = row["City Or County"].strip()
   if state + ":" + city not in incidents:
      incidents[state + ":" + city] = int(row["# Injured"]) + int(row["#
          Killed"])
   else:
      incidents[state + ":" + city] += int(row["# Injured"]) + int(row["#
          Killed"])
   df = df.drop([i])

df = df.drop(columns = ["# Killed", "# Injured"])
for k, v in incidents.iteritems():
   key = k.split(':')
   city = key[1]
   state = key[0]
   numberOfDeaths = v
   df = df.append({"State" : state, "City Or County" : city, "# Incidents" :
       int(numberOfDeaths)}, ignore_index = True)

return df
```

4.2.2 Klasifikacija u Knime-u

Prvo se u CSV Reader-u citaju podaci koji su generisani, to se prosledjuje u Rule Engine gde se vrednosti kolone # Incidents menjaju i dobijaju vrednosti Low i High u zavisnosti od toga koja im je vrednost. Onda se ide u Partitioning gde se podaci na slucajan nacin dele na trening i test podatke (33% su trening podaci). Test podaci se salju u Decisiion Tree Predictor, a trening u Decisiion Tree Learner ciji se izlaz prosledjuje u Predictor. Predictor salje svoj izlaz u Scorer odakle mozemo videti matricu konfuzije, gresku, preciznost... a u Predictor-u mozemo videti drvo odlucivanja.

Greska je oko 30%, preciznost oko 70%.

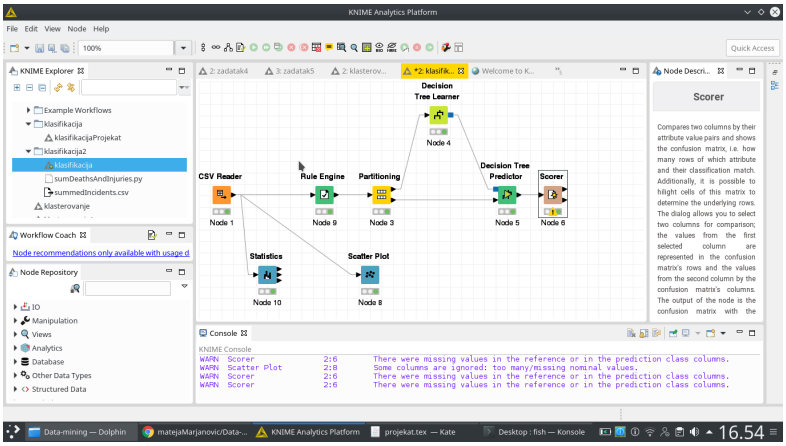

Figure 10: Izlged klasifikacije u Knime-u

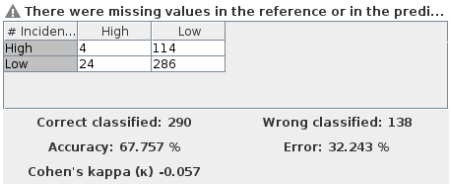

Figure 11: Matrica konfuzije