FinalProjectCode

September 15, 2019

1 1. Create the Database and Proper Schema

[1]:
```
##Create one database with three tables


import sqlite3
db = sqlite3.connect('alien-life.db')
cursor = db.cursor()

cursor.execute("PRAGMA foreign_keys = 1")

cross_ref_table = """CREATE TABLE CROSS_REF_TABLE(
            patient_id      INTEGER PRIMARY KEY AUTOINCREMENT,
            patient_age     FLOAT
);"""

cursor.execute(cross_ref_table)
db.commit()

boran_table = """CREATE TABLE BORAN_TABLE(
            patient_id      INTEGER NOT NULL,
            blood_pressure  FLOAT,
            exercise        FLOAT,
            weight          FLOAT,
            glucose         FLOAT,
            bmi             FLOAT,
            planet_id       INTEGER,
            FOREIGN KEY(patient_id) REFERENCES cross_ref_table(patient_id)
);"""

cursor.execute(boran_table)
db.commit()

radan_table = """CREATE TABLE RADAN_TABLE(
            patient_id      INTEGER NOT NULL,
            blood_pressure  FLOAT,
```

```
            exercise        FLOAT,
            weight          FLOAT,
            glucose         FLOAT,
            bmi             FLOAT,
            planet_id       INTEGER,
            FOREIGN KEY(patient_id) REFERENCES cross_ref_table(patient_id)
);"""

cursor.execute(radan_table)

db.commit()

db.close()
```

[2]:
```
#Third, CROSS_REF_TABLE
import sqlite3
import csv

db     = sqlite3.connect('alien-life.db')
cursor = db.cursor()
cursor.execute("PRAGMA foreign_keys = 1")

with open('datasets/deidentify_list_cross_ref.csv', 'r') as f:

    fcsv = csv.reader(f)

    recs_to_load = [record for record in fcsv]

cursor.executemany("INSERT INTO CROSS_REF_TABLE VALUES (?, ?)", recs_to_load[1:
 ↪])

db.commit()
db.close()
```

[3]:
```
##read the csv files into the tables respectively
#First, BORAN_TABLE
import sqlite3
import csv

db     = sqlite3.connect('alien-life.db')
cursor = db.cursor()
cursor.execute("PRAGMA foreign_keys = 1")

with open('datasets/boran.csv', 'r') as f:

    fcsv = csv.reader(f)

    recs_to_load = [record for record in fcsv]
```

```
cursor.executemany("INSERT INTO BORAN_TABLE VALUES (?, ?, ?, ?, ?, ?, ?)",␣
 ↪recs_to_load[1:])

db.commit()
db.close()
```

[4]:
```
#Second, RADAN_TABLE
import sqlite3
import csv

db     = sqlite3.connect('alien-life.db')
cursor = db.cursor()
cursor.execute("PRAGMA foreign_keys = 1")

with open('datasets/radan.csv', 'r') as f:

    fcsv = csv.reader(f)

    recs_to_load = [record for record in fcsv]

cursor.executemany("INSERT INTO RADAN_TABLE VALUES (?, ?, ?, ?, ?, ?, ?)",␣
 ↪recs_to_load[1:])

db.commit()
db.close()
```

2 Check the Consisitency and Validity of the Database

[5]:
```
##Here we check that the db enforces the appropriate validity checks for the␣
 ↪boran_table
import pandas as pd
import sqlite3

db = sqlite3.connect('alien-life.db')
cursor = db.cursor()
cursor.execute("PRAGMA foreign_keys = 1")

cursor.execute("UPDATE boran_table SET patient_id = 3 WHERE exercise > 143")

db.commit()
db.close()
```

```
      ␣
↪-------------------------------------------------------------------------
```

```
       IntegrityError                            Traceback (most recent call␣
↪last)

       <ipython-input-5-f853f751c449> in <module>
         7 cursor.execute("PRAGMA foreign_keys = 1")
         8
   ----> 9 cursor.execute("UPDATE boran_table SET patient_id = 3 WHERE exercise␣
↪> 143")
        10
        11 db.commit()


       IntegrityError: FOREIGN KEY constraint failed
```

[6]:
```
##Here we check that the db enforces the appropriate validity checks for the␣
 ↪radan_table
import pandas as pd
import sqlite3

db = sqlite3.connect('alien-life.db')
cursor = db.cursor()
cursor.execute("PRAGMA foreign_keys = 1")

cursor.execute("UPDATE radan_table SET patient_id = 4569 WHERE exercise > 15")

db.commit()
db.close()
```

```
      ␣
↪---------------------------------------------------------------------------

       IntegrityError                            Traceback (most recent call␣
↪last)

       <ipython-input-6-82c0e6ec7607> in <module>
         7 cursor.execute("PRAGMA foreign_keys = 1")
         8
   ----> 9 cursor.execute("UPDATE radan_table SET patient_id = 4569 WHERE␣
↪exercise > 15")
        10
        11 db.commit()


       IntegrityError: FOREIGN KEY constraint failed
```

3 Boran Linear Regression Process

[7]:
```
import pandas as pd
import sqlite3

db     = sqlite3.connect('alien-life.db')


df = pd.read_sql_query("""SELECT boran_table.patient_id, boran_table.
 ↪blood_pressure, cross_ref_table.patient_age, boran_table.exercise,␣
 ↪boran_table.weight, boran_table.glucose, boran_table.bmi
       FROM boran_table
       LEFT JOIN cross_ref_table ON boran_table.patient_id = cross_ref_table.
 ↪patient_id""", db)

##there exists no null values, but in case the full data set has them we␣
 ↪include the following
df.fillna(df.mean()).head(5)
```

[7]:

	patient_id	blood_pressure	patient_age	exercise	weight	glucose	bmi
0	83944	199.378675	86.055427	143.205239	148.036310	125.230960	1.045336
1	41989	191.853108	76.957462	102.544295	141.588777	119.807890	0.988849
2	94365	207.226606	85.005850	36.446482	157.499291	131.750089	1.095599
3	93464	203.442508	85.762916	91.476012	152.871206	128.586136	1.071569
4	57985	194.236774	72.326139	9.071701	145.584071	122.141463	1.000579

[8]:
```
cols = ['patient_age']
X_boran = df[cols]
Y_boran = df['blood_pressure']
```

[9]:
```
from sklearn.linear_model import LinearRegression
```

[10]:
```
model = LinearRegression()
```

[11]:
```
model.fit(X_boran, Y_boran)
```

[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[12]:
```
##find the mk value
model.coef_
```

[12]:
```
array([1.14074587])
```

[13]:
```
##find the Bk value
model.intercept_
```

[13]:
```
106.45741672344948
```

[14]:
```
##test accuracy
preds = model.predict(X_boran)
preds.std()
error = preds - Y_boran

total_error = 0
for item in error:
    total_error += (item * item)

print('total_error: ', total_error)

mse = total_error/len(error)
print('mse: ', mse)

import numpy as np
rmse = np.sqrt(mse)
print('rmse: ', rmse)
```

```
total_error:  636.396934835724
mse:  21.213231161190798
rmse:  4.605782361465943
```

4 Radan Linear Regression Process

[15]:
```
import pandas as pd
import sqlite3

db     = sqlite3.connect('alien-life.db')


df2 = pd.read_sql_query("""SELECT radan_table.patient_id, radan_table.
 ↪blood_pressure, cross_ref_table.patient_age, radan_table.exercise,␣
 ↪radan_table.weight, radan_table.glucose, radan_table.bmi
        FROM radan_table
        LEFT JOIN cross_ref_table ON radan_table.patient_id = cross_ref_table.
 ↪patient_id""", db)

##there exists no null values, but in case the full data set has them we␣
 ↪include the following
df.fillna(df.mean()).head(1)
```

[15]:
```
   patient_id  blood_pressure  patient_age    exercise      weight    glucose  \
0       83944      199.378675    86.055427  143.205239  148.03631  125.23096

         bmi
0  1.045336
```

[16]:
```
cols = ['patient_age']
X_radan = df2[cols]
Y_radan = df2['blood_pressure']
```

[17]:
```
from sklearn.linear_model import LinearRegression
```

[18]:
```
model = LinearRegression()
```

[19]:
```
model.fit(X_radan, Y_radan)
```

[19]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[20]:
```
##find the mk value
model.coef_
```

[20]:
```
array([0.47088464])
```

[21]:
```
##find the Bk value
model.intercept_
```

[21]:
```
63.825776453326256
```

[22]:
```
##test accuracy
preds = model.predict(X_radan)
preds.std()
error = preds - Y_radan

total_error = 0
for item in error:
    total_error += (item * item)

print('total_error: ', total_error)

mse = total_error/len(error)
print('mse: ', mse)

import numpy as np
rmse = np.sqrt(mse)
print('rmse: ', rmse)
```

```
total_error:  216.22906224639857
mse:  7.207635408213286
rmse:  2.6847039703127953
```

5 Generate plots for the histogram age distribution for each planet

5.0.1 1. Boran

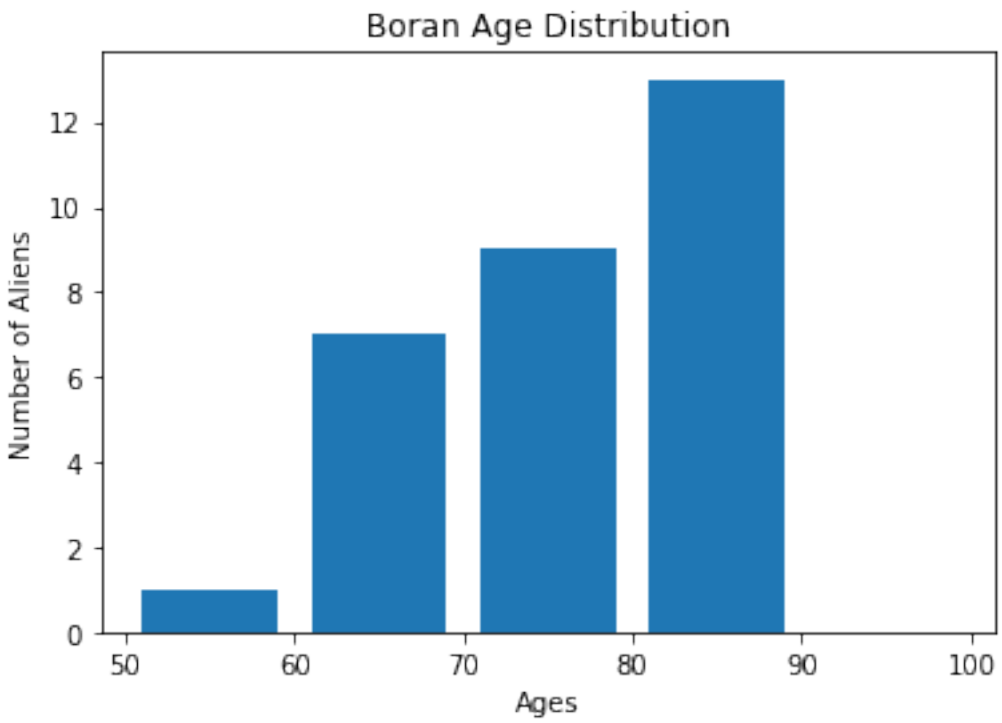
[23]:
```
import numpy as np
bages = np.asarray(X_boran)
```

[33]:
```
import matplotlib.pyplot as plt

bins = [50, 60, 70, 80, 90, 100]

plt.hist(bages, bins, histtype='bar', rwidth = 0.8)

plt.xlabel('Ages')
plt.ylabel('Number of Aliens')
plt.title('Boran Age Distribution')
plt.show()
```



5.0.2 (a) What is the mean (average) life expectance of the creatures on Boran?

[26]:
```
bages.mean()
```

[26]:
```
76.41943114989894
```

5.0.3 (b) What is the probability of a creature living past the mean life expectancy on Boran?

[27]:
```
count = 0
for age in bages:
    if age > bages.mean():
        count += 1
prob = count/len(bages)
prob
```

[27]:
```
0.5
```

5.0.4 2. Radan

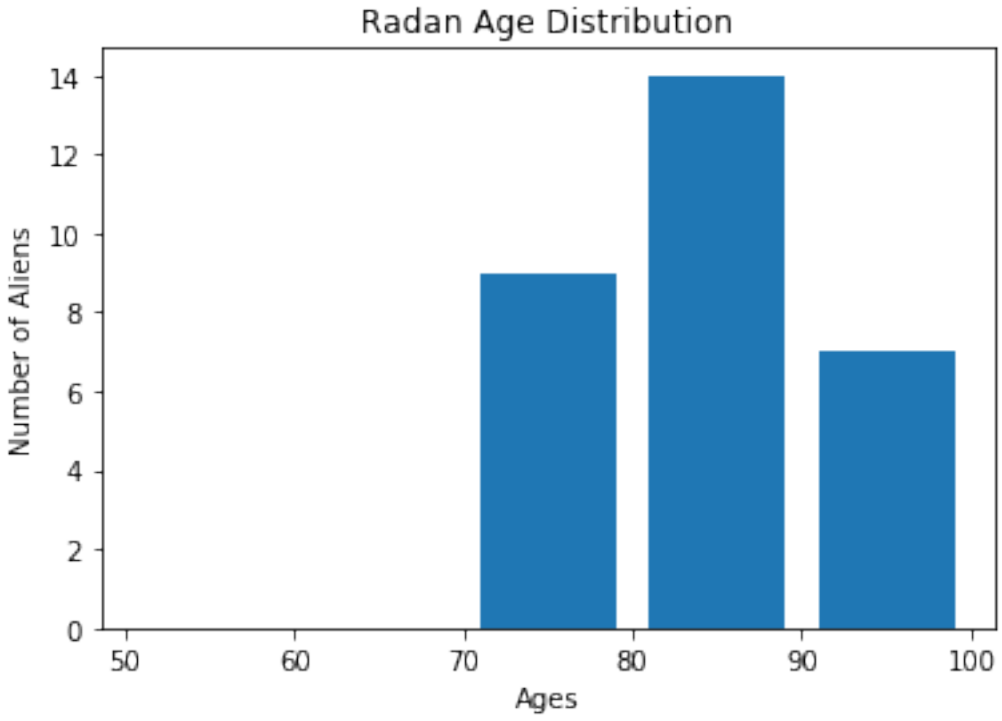
[28]:
```
import numpy as np
rages = np.asarray(X_radan)
```

[32]:
```
import matplotlib.pyplot as plt

bins = [50, 60, 70, 80, 90, 100]

plt.hist(rages, bins, histtype='bar', rwidth = 0.8)

plt.xlabel('Ages')
plt.ylabel('Number of Aliens')
plt.title('Radan Age Distribution')
plt.show()
```



5.0.5 (a) What is the mean (average) life expectance of the creatures on Radan?

[30]:
```
rages.mean()
```

[30]:
```
84.21706879853835
```

5.0.6 b) What is the probability of a creature living past the mean life expectancy on Boran?

[31]:
```
count = 0
for age in rages:
    if age > rages.mean():
        count += 1
prob = count/len(rages)
prob
```

[31]:
```
0.43333333333333335
```