Μέλος 1ο : Βακαλόπουλος Δημήτριος
ΑΜ:1059564

Μέλος 2ο : Ρουμελιώτης Νικόλαος
ΑΜ: 1047174

Ερώτημα 1

Α) Αρχικά εισάγουμε όλες τις απαραίτητες βιβλιοθήκες.

```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import cm
import seaborn as sns
import math
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

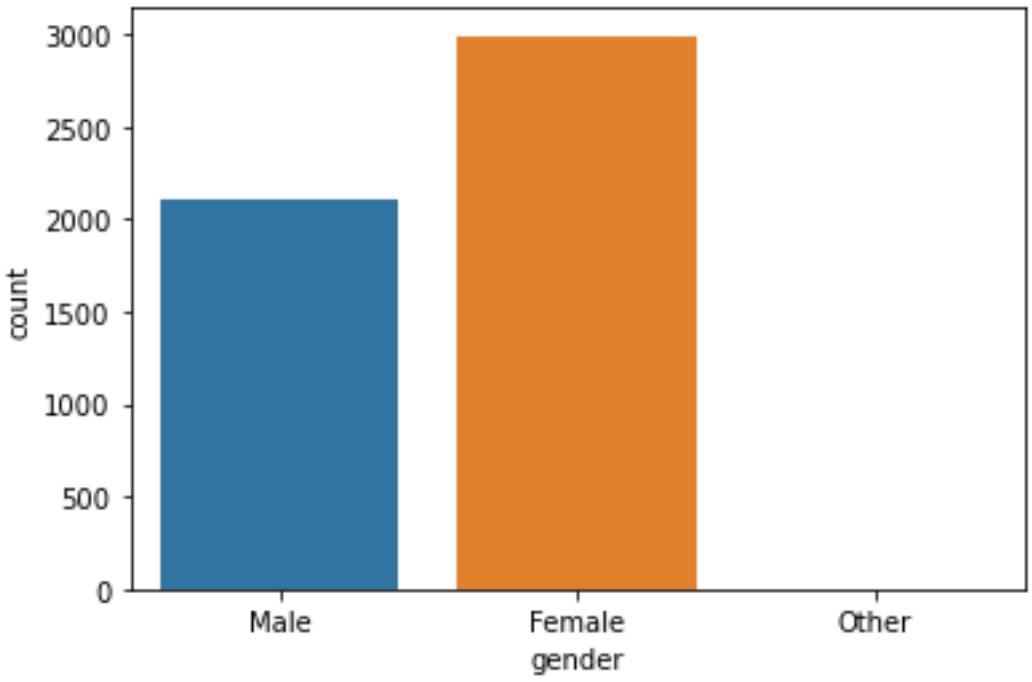
‘Επειτα γίνεται ανάγνωση του αρχείου csv που χρειαζόμαστε.

```
mydata = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

Εξάγουμε το γράφημα εκείνο που απεικονίζει πόσους άνδρες έχουμε και πόσες γυναίκες με τις παρακάτω εντολές

```
sns.countplot(mydata['gender'],label="Count")
plt.show()
```

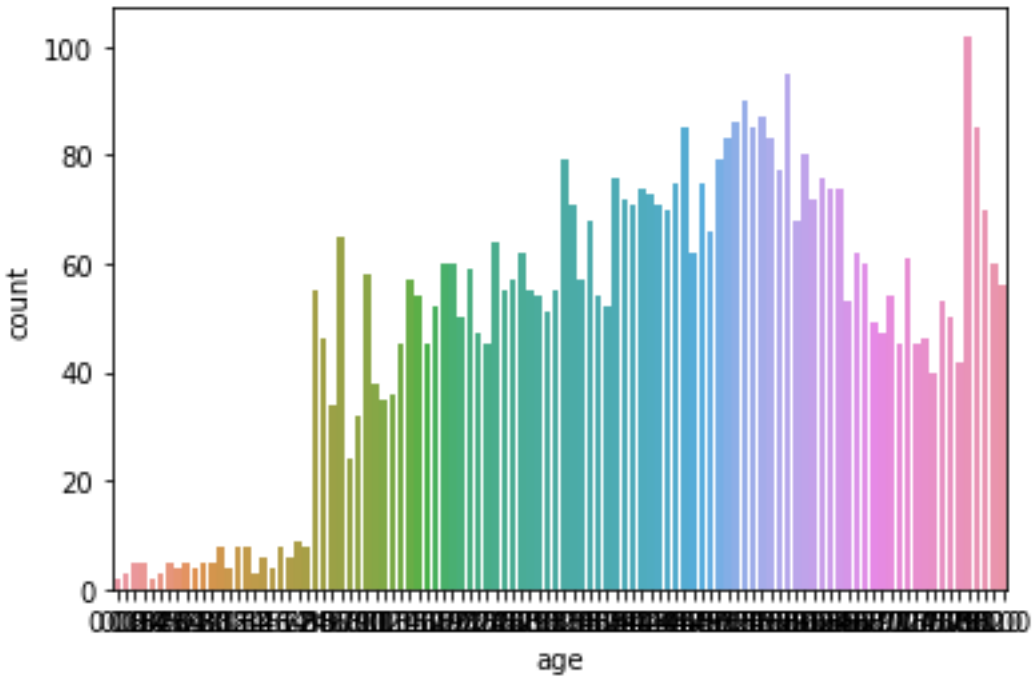
Παρακάτω έχουμε το γράφημα :

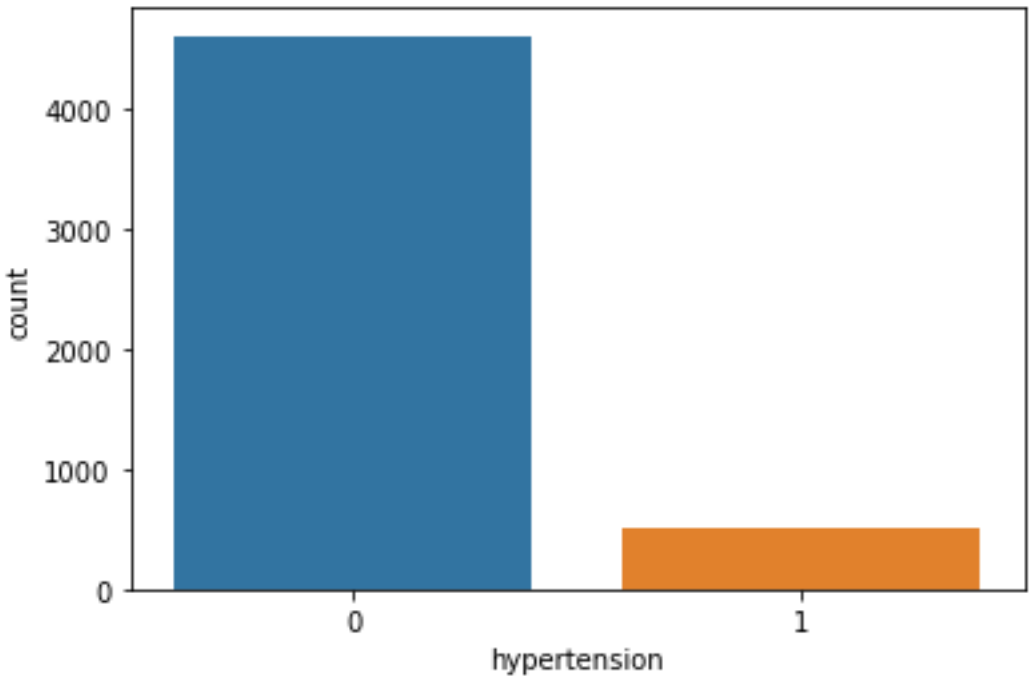


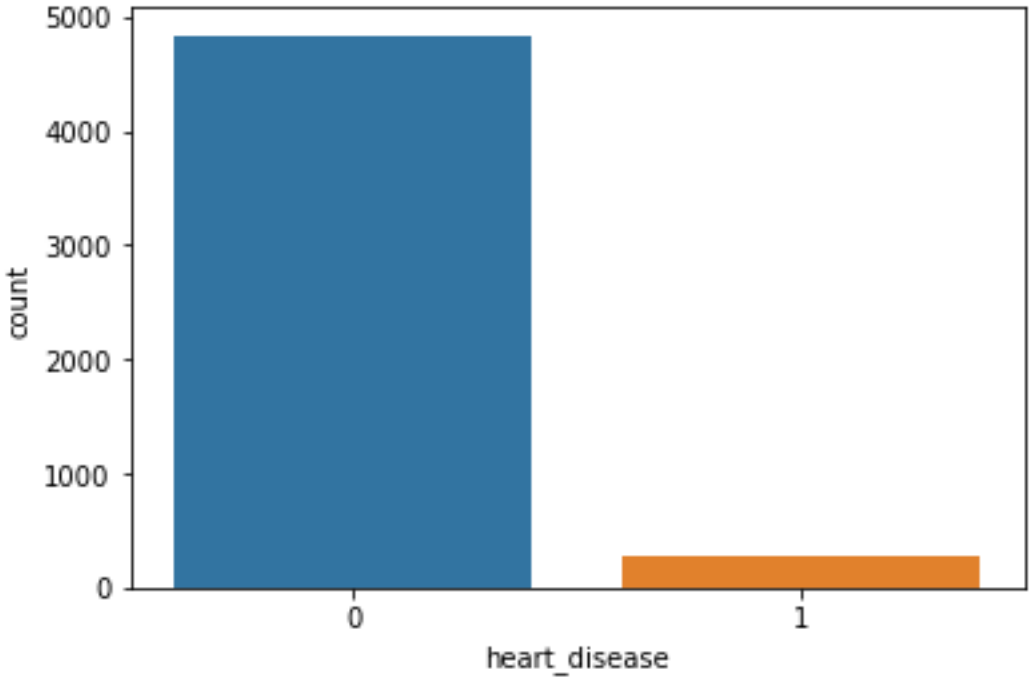
Ομοίως και με τις παρακάτω εντολές:

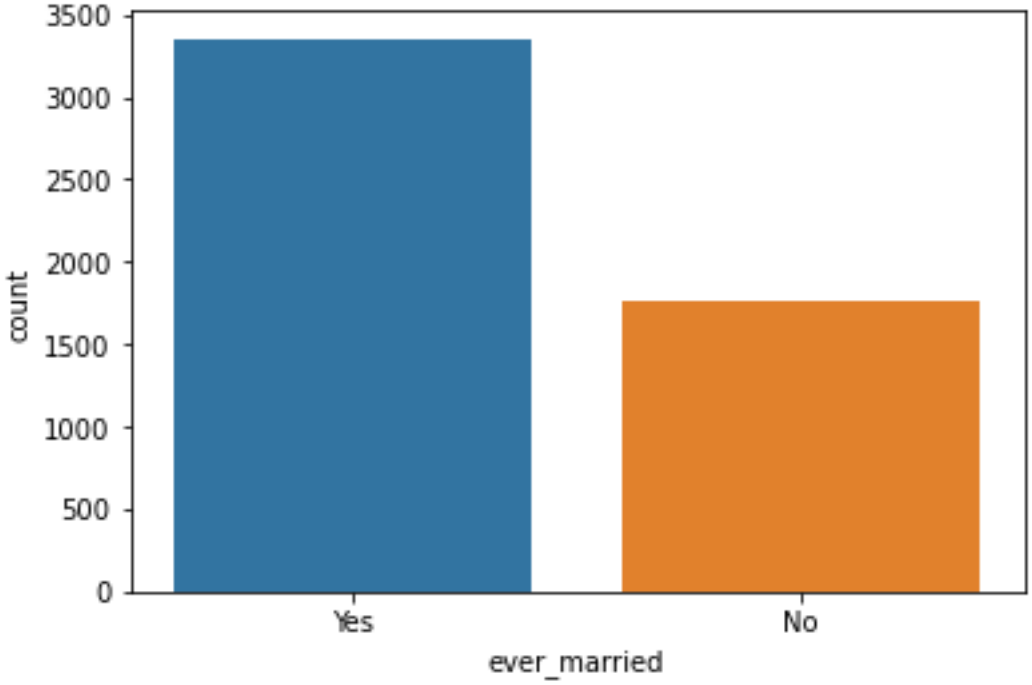
```
sns.countplot(mydata['age'],label="Count")
plt.show()

sns.countplot(mydata['hypertension'],label="Count")
plt.show()

sns.countplot(mydata['heart_disease'],label="Count")
plt.show()

sns.countplot(mydata['ever_married'],label="Count")
plt.show()

sns.countplot(mydata['work_type'],label="Count")
plt.show()

sns.countplot(mydata['Residence_type'],label="Count")
plt.show()

sns.countplot(mydata['avg_glucose_level'],label="Count")
plt.show()

sns.countplot(mydata['bmi'],label="Count")
plt.show()

sns.countplot(mydata['smoking_status'],label="Count")
plt.show()
```

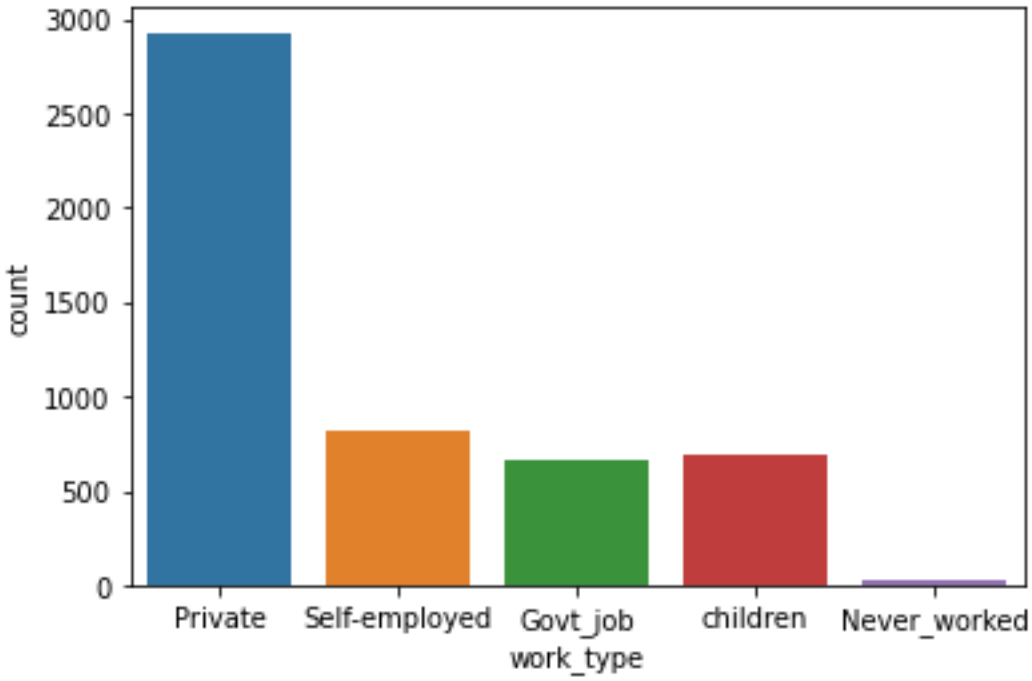
```
sns.countplot(mydata['stroke'],label="Count")
plt.show()
```

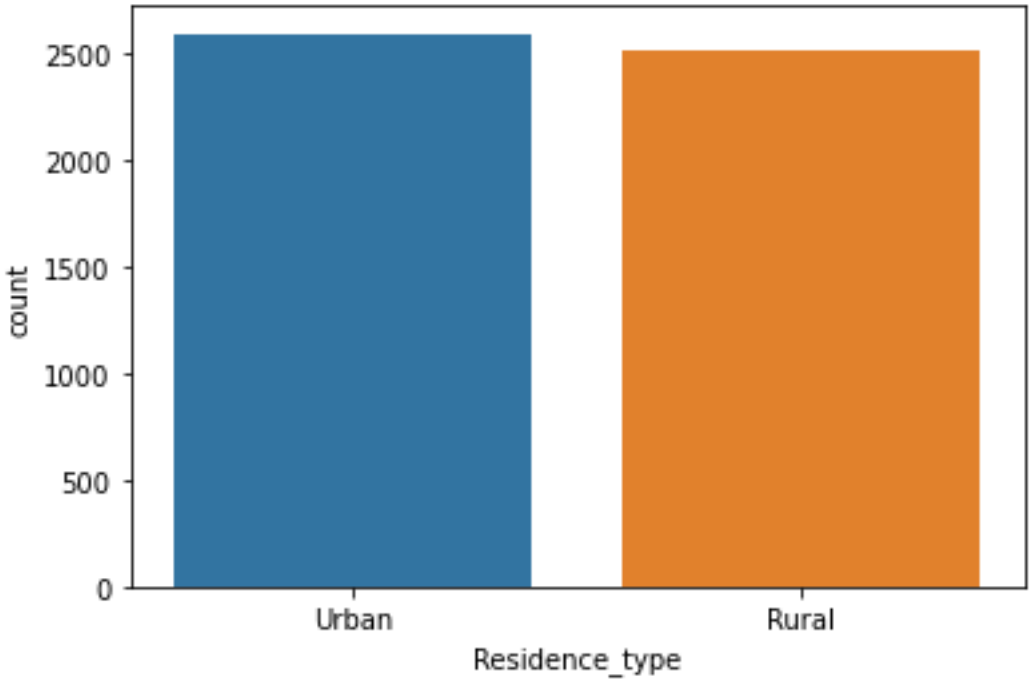

100
80
60
40
20
0
count
age

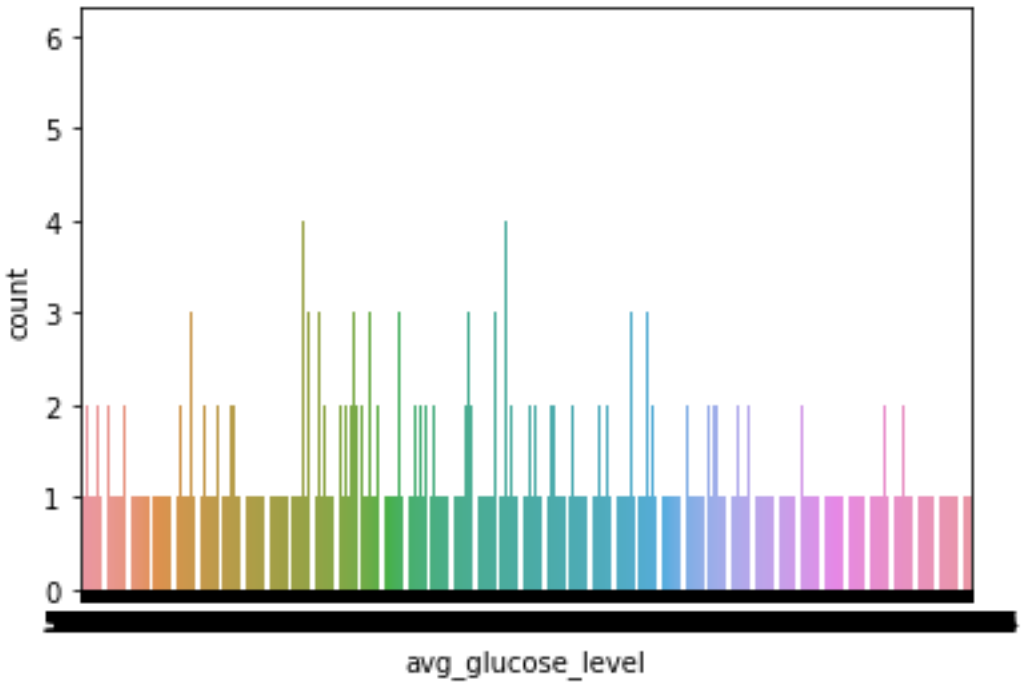

4000
3000
2000
1000
0
count
0
1
hypertension

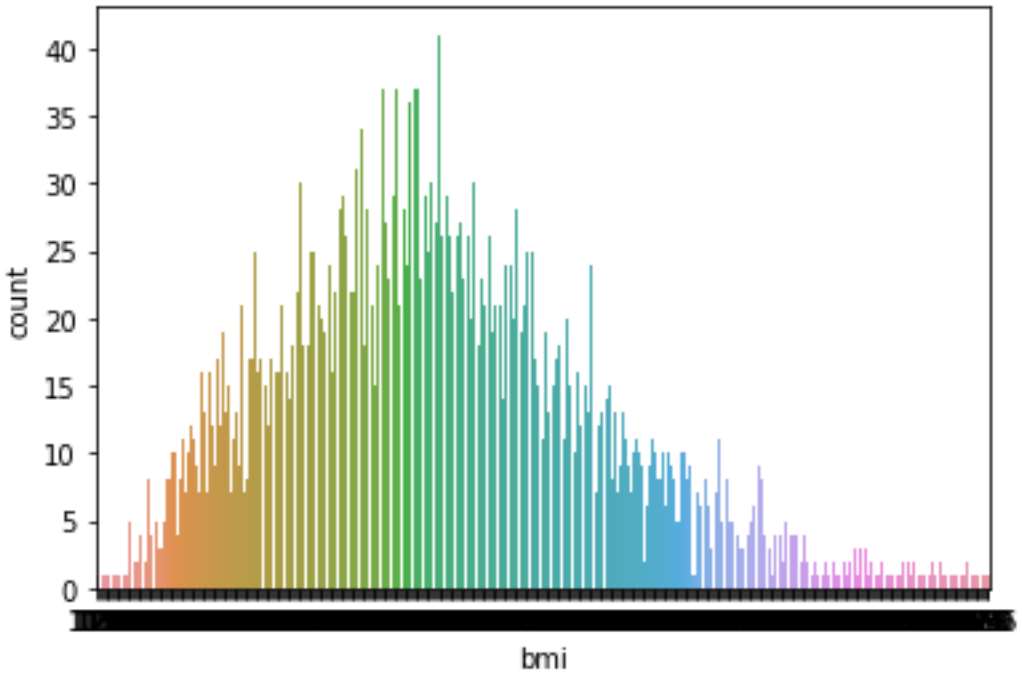

5000
4000
3000
2000
1000
0
count
0
1
heart_disease

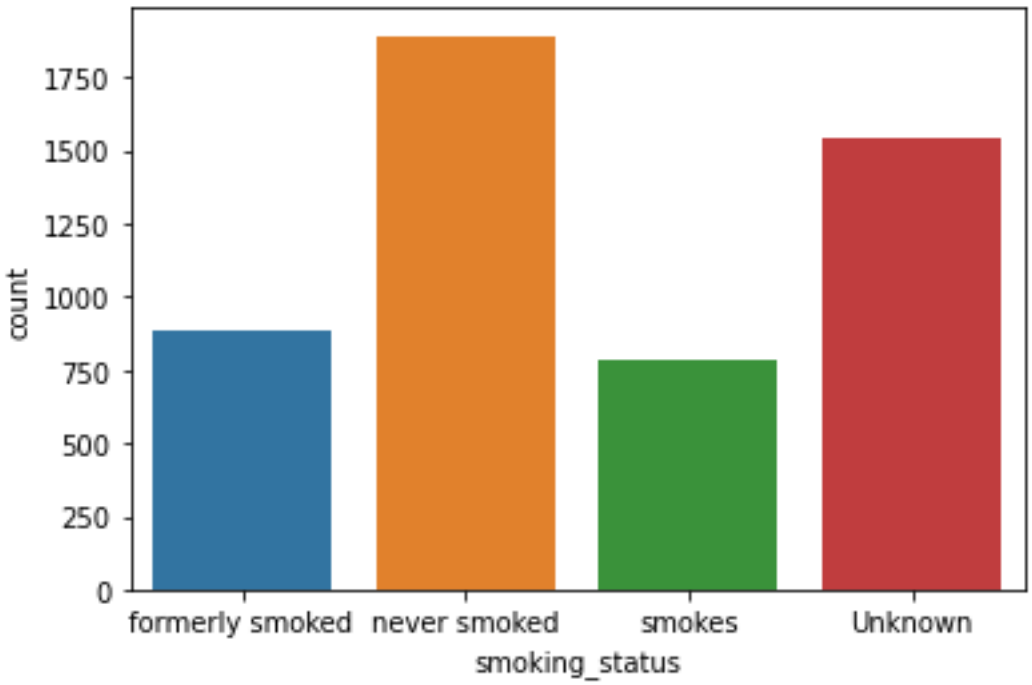

3500
3000
2500
2000
1500
1000
500
0
count
Yes
No
ever_married

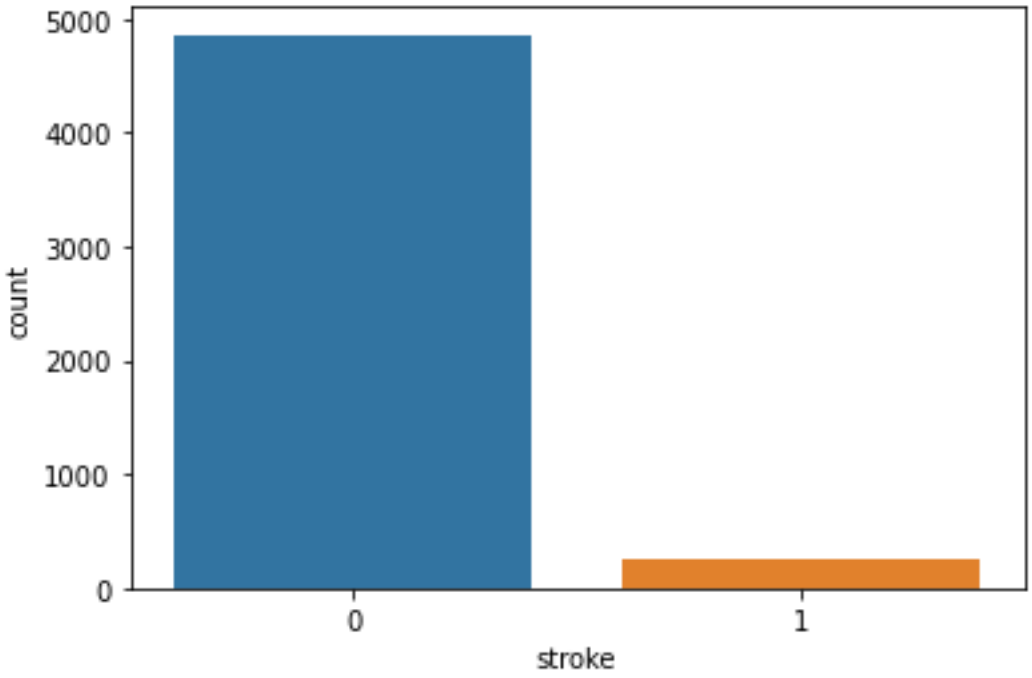

3000
2500
2000
1500
1000
500
0
count
Private
Self-employed
Govt_job
children
Never_worked
work_type


2500
2000
1500
1000
500
0
count
Urban
Rural
Residence_type


count
avg_glucose_level


count
bmi


1750
1500
1250
1000
750
500
250
0
count
formerly smoked
never smoked
smokes
Unknown
smoking_status


5000
4000
3000
2000
1000
0
count
0
1
stroke

Β1,Γ) Στο συγκεκριμένο ερώτημα στόχος μας είναι να εντοπίσουμε καθώς και να χειριστούμε τις ελλιπείς τιμές με τη μέθοδο της αφαίρεσης στήλης. Στη προκειμένη περίπτωση έχουμε μερικές τιμές του bmi που είναι ελλειπείς, οπότε αφαιρούμε τη στήλη bmi.

Αρχικά εισάγουμε όλες τις απαραίτητες βιβλιοθήκες

```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import cm
import seaborn as sns
import math
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

Έπειτα γίνεται ανάγνωση του dataset που χρειαζόμαστε

```
mydata = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

Αποθηκεύουμε στη λίστα ‘’feature_names” όλες τις στήλες εκτός από τη στήλη ‘’bmi’’ και “stroke’’.

```
feature_names = ['gender', 'age', 'hypertension',
```

Η μεταβλητή X είναι η είσοδος μας, η οποία περιέχει όλες τις στήλες εκτός από το bmi και το stroke , και η μεταβλητή y είναι η έξοδος μας, που είναι η στήλη stroke.

```
X = mydata[feature_names]

y = mydata['stroke']
```

Εφαρμόζουμε labelencoder

```
le = LabelEncoder()

X['gender']= le.fit_transform(X['gender'])

X['ever_married']= le.fit_transform(X['ever_married'])

X['work_type']= le.fit_transform(X['work_type'])

X['Residence_type']= le.fit_transform(X['Residence_type'])

X['smoking_status']= le.fit_transform(X['smoking_status'])
```

Χωρίζουμε το dataset σε training-test με αναλογία 75%-25%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

Δημιουργούμε το μοντέλο randomForest

```
clf=RandomForestClassifier(n_estimators=5000)
```

Εκπαιδεύσουμε το μοντέλο

```
clf.fit(X_train,y_train)
```

Τεστάρουμε το μοντέλο

```
y_pred=clf.predict(X_test)
```

Υπολογίζουμε την απόδοση του μοντέλου χρησιμοποιώντας μετρικές f1 score, precision και recall

```
precision = precision_score(y_test, y_pred, average='binary')

print('Precision: %.3f' % precision)

recall = recall_score(y_test, y_pred, average='binary')
print('Recall: %.3f' % recall)

score = f1_score(y_test, y_pred, average='binary')
print('F-Measure: %.3f' % score)
```

Παρακάτω έχουμε την απόδοση του μοντέλου

```
Precision: 0.250
Recall: 0.015
F-Measure: 0.028
```

Β2,Γ) Στο συγκεκριμένο ερώτημα στόχος μας είναι να εντοπίσουμε καθώς και να αντικαταστήσουμε τις ελλιπείς τιμές με το μέσο όρο των στοιχείων της στήλης bmi.

Αρχικά λειτουργούμε ομοίως με το προηγούμενο ερώτημα μόνο που στη προκειμένη περίπτωση στη λίστα ‘’feature_names’’ έχουμε και τη στήλη ‘’bmi’’.

```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import cm
import seaborn as sns
import math
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

```
mydata = pd.read_csv('healthcare-dataset-stroke-data.csv')


feature_names = ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married','work_ty
X = mydata[feature_names]

y = mydata['stroke']

#efarmogi labelencoder

le = LabelEncoder()

X['gender']= le.fit_transform(X['gender'])

X['ever_married']= le.fit_transform(X['ever_married'])

X['work_type']= le.fit_transform(X['work_type'])

X['Residence_type']= le.fit_transform(X['Residence_type'])

X['smoking_status']= le.fit_transform(X['smoking_status'])
```

Υπολογίζουμε τον μέσο όρο των τιμών της στήλης bmi

```
sum_bmi=0

count_bmi=0

for x in X['bmi']:
    if  not math.isnan(x):
        sum_bmi=sum_bmi+x
        count_bmi=count_bmi+1


avg_bmi = sum_bmi/count_bmi
```

Αντικαθιστούμε τις ελλειπείς τιμές της στήλης bmi με το μέσο όρο των τιμών της στήλης bmi

```
for i in range(len(X['bmi'])):
    if math.isnan(X['bmi'][i]):
        X['bmi'][i]=avg_bmi
```

Χωρίζουμε το dataset σε training-test με αναλογία 75%-25%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

Δημιουργούμε το μοντέλο randomForest

```
clf=RandomForestClassifier(n_estimators=6000)
```

Εκπαιδεύσουμε το μοντέλο

```
clf.fit(X_train,y_train)
```

```
y_pred=clf.predict(X_test)
```

Τεστάρουμε το μοντέλο

```
y_pred=clf.predict(X_test)
```

Υπολογίζουμε την απόδοση του μοντέλου χρησιμοποιώντας μετρικές f1 score, precision και recall

```
precision = precision_score(y_test, y_pred, average='binary')

print('Precision: %.3f' % precision)

recall = recall_score(y_test, y_pred, average='binary')
print('Recall: %.3f' % recall)

score = f1_score(y_test, y_pred, average='binary')
print('F-Measure: %.3f' % score)
```

Παρακάτω έχουμε τον μέσο όρο των τιμών της στήλης bmi καθώς και τα αποτελέσματα της απόδοσης του μοντέλου

```
28.893236911794673
Precision: 0.200
Recall: 0.018
F-Measure: 0.033
```

Β3,Γ) Στο συγκεκριμένο ερώτημα στόχος μας είναι να εντοπίσουμε καθώς και να χειριστούμε τις ελλιπείς τιμές με τη μέθοδο linear regression και να αντικαταστήσουμε τις ελλειπείς τιμές με τις νέες.

Αρχικά λειτουργούμε ομοίως με το προηγούμενο ερώτημα

```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import cm
import seaborn as sns
import math
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.linear_model import LinearRegression
```

```
mydata = pd.read_csv('healthcare-dataset-stroke-data.csv')


feature_names = ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married','work_ty
X = mydata[feature_names]

y = mydata['stroke']

#efarmogi labelencoder

le = LabelEncoder()

X['gender']= le.fit_transform(X['gender'])

X['ever_married']= le.fit_transform(X['ever_married'])

X['work_type']= le.fit_transform(X['work_type'])

X['Residence_type']= le.fit_transform(X['Residence_type'])

X['smoking_status']= le.fit_transform(X['smoking_status'])
```

Εδώ κάνουμε ότι χρειάζεται για το linear regression

```
feature_names_reg = ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married','wo
X_reg = mydata[feature_names_reg]

y_reg = mydata['bmi']

#efarmogi labelencoder

le_reg = LabelEncoder()

X_reg['gender']= le_reg.fit_transform(X_reg['gender'])

X_reg['ever_married']= le_reg.fit_transform(X_reg['ever_married'])

X_reg['work_type']= le_reg.fit_transform(X_reg['work_type'])

X_reg['Residence_type']= le_reg.fit_transform(X_reg['Residence_type'])

X_reg['smoking_status']= le_reg.fit_transform(X_reg['smoking_status'])


X_reg_train = pd.DataFrame(columns=feature_names_reg,index=range(4909))
y_reg_train=pd.Series()
```

Επιλέγουμε εκείνες τις εγγραφές οι οποίες δεν έχουν naN στο bmi. Αυτές είναι οι εγγραφές οι οποίες θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου μας

```
count=0

for i in range(len(X_reg)):
    if not np.isnan(y_reg[i]):
        X_reg_train['gender'].loc[count] = X_reg['gender'].loc[i]
        X_reg_train['age'].loc[count] = X_reg['age'].loc[i]
        X_reg_train['hypertension'].loc[count] = X_reg['hypertension'].loc[i]
        X_reg_train['heart_disease'].loc[count] = X_reg['heart_disease'].loc[i]
        X_reg_train['ever_married'].loc[count]=X_reg['ever_married'].loc[i]
        X_reg_train['work_type'].loc[count]=X_reg['work_type'].loc[i]
        X_reg_train['Residence_type'].loc[count] = X_reg['Residence_type'].loc[i]
        X_reg_train['avg_glucose_level'].loc[count]=X_reg['avg_glucose_level'].loc[i]
        X_reg_train['smoking_status'].loc[count]=X_reg['smoking_status'].loc[i]

        y_reg_train.loc[count] = y_reg.loc[i]
        count=count+1
```

Δημιουργούμε το μοντέλο linear Regression

```
regressor.fit(X_reg_train, y_reg_train)
```

Εκπαιδεύουμε το μοντέλο linear Regression

```
regressor = LinearRegression()
```

Διατρέχουμε το dataset

```
for i in range(len(X)):
    #vriskoume tis eggrafes opu exoun nan stin timi tou bmi
    if np.isnan(X['bmi'].loc[i]):
        #dedomena pou tha dosoume sto logistic regression gia na ginoun predict
        myinput = X_reg.loc[i]
        #ginetai o katallilos metasximatismos gia na mporesei na epiteyxuei to prediction
        myinput=myinput.values.reshape(1,-1)
        #edo simplironetai i timi pou leipei me tin timi pou ginetai predict
        X['bmi'].loc[i] = regressor.predict(myinput)
```

Χωρίζουμε το dataset σε training-test με αναλογία 75%-25%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

Δημιουργούμε το μοντέλο Random Forest

```
clf=RandomForestClassifier(n_estimators=6000)
```

Εκπαιδεύουμε το μοντέλο

```
clf.fit(X_train,y_train)
```

Τεστάρουμε το μοντέλο

```
y_pred=clf.predict(X_test)
```

Υπολογίζουμε την απόδοση του μοντέλου χρησιμοποιώντας μετρικές f1 score, precision και recall

```
precision = precision_score(y_test, y_pred, average='binary')

print('Precision: %.3f' % precision)

recall = recall_score(y_test, y_pred, average='binary')
print('Recall: %.3f' % recall)

score = f1_score(y_test, y_pred, average='binary')
print('F-Measure: %.3f' % score)
```

Παρακάτω έχουμε τα αποτελέσματα της απόδοσης του μοντέλου

```
Precision: 0.500
Recall: 0.018
F-Measure: 0.035
```

Β4-Γ)Στο συγκεκριμένο ερώτημα εφαρμόζουμε τον αλγόριθμο KNN όπου λειτουργούμε ακριβώς με τον ίδιο τρόπο που λειτουργήσαμε και στο Β3-Γ ερώτημα μόνο που στη προκειμένη περίπτωση έχουμε KNN.

Αντί για αυτό το μοντέλο

```
regressor = LinearRegression()
```

Έχουμε:

```
classifier = KNeighborsClassifier(n_neighbors=5)
```

Παρακάτω έχουμε τα αποτελέσματα της απόδοσης του μοντέλου

```
Precision: 0.500
Recall: 0.015
F-Measure: 0.029
```

Ερώτημα 2

Αρχικά εισάγουμε όλες τις απαραίτητες βιβλιοθήκες

```
import pandas as pd
import numpy as np
import tensorflow.keras as keras
from gensim.models import word2vec
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

Διαβάζουμε το αρχείο που θέλουμε

```
mydata = pd.read_csv('spam_or_not_spam.csv')
```

Στη λίστα ‘’emails’’ κρατάμε μόνο τα emails, στη λίστα output κρατάμε τις τιμές 0,1 που αντιπροσωπεύουν για το αν είναι spam ή όχι. Ορίζουμε και λίστα ‘’email_split’’.

```
emails= mydata['email']

output=mydata['label']

email_split=[]
```

Για κάθε email κάνουμε split και δημιουργούμε μια λίστα από τις λέξεις από τις οποίες αποτελείται

```
for email in emails:
    email_split.append(str(email).split())
```

Εφαρμόζουμε το μοντέλο word2vec για να μετατρέψουμε τις λέξεις σε διανύσματα

```
model = word2vec.Word2Vec(email_split, min_count=1)
```

```
vectorized_emails=[]

#gia kathe stoixeio tis listas email_split(poy einai mia lista leksewn)
for x in email_split:
   #i lista me ta dyanismata gia kathe email
    vector_email  = []
    #gia kathe leksi tou emamil
    for y in x:
      #prosthetoume to kodikopoihmeno dianisma(poy proekypse apo tin texniki word embedin
        vector_email.append(model.wv[y])
    #prosthetoume stin megali lista, tin lista twn dianismatwn poy antistoixei sto email
    vectorized_emails.append(vector_email)

final_emails=[]
```

Προσθέτουμε στον ‘’final_emails’’ τον μέσο όρο των διανυσμάτων των λέξεων που αποτελούν το email (επειδή κάθε email αποτελείται από πολλά διανύσματα, επειδή θέλουμε να έχουμε τελικά ένα διάνυσμα ανά email,παίρνουμε τον μέσο όρο των διανυσμάτων των λέξεων που τον αποτελούν)

```
for x in vectorized_emails:

    final_emails.append(np.average(x, axis=0))
```

Γίνεται μετασχηματισμός των δεδομένων σε κατάλληλη μορφή έτσι ώστε να μπορούν να εισαχθούν στο νευρωνικό δίκτυο

```
#lista me tis eisidous
final_input=[]

#lista me tis eksodous
final_output=[]

#gia kathe email(to opoio exei ypostei dianismatopoihsi)
for i in range(len(final_emails)):
    #an den einai nAN
    if not np.isnan(final_emails[i]).all():
        #dimiourgoume mia prosoroni lista
        temp_list=[]
        #gia kathe dianisma pou einai mesa sto email
        for x in final_emails[i]:
            #to prosthetoume stin prosorini lista
            temp_list.append(x)
        #prosthetoume tin prosorini lista stin teliki (gia tis eisodous)
        final_input.append(temp_list)
        #prostetoume tin eksodo stin teliki lista (gia tis eksodous )
        final_output.append(output[i])

#ara to final_input einai mia lista apo listes

#kai to final_output einai mia lista
```

Μετατρέπουμε την λίστα με τα δεδομένα εισόδου σε 2d array έτσι ώστε να μπορεί να εισαχθεί στο νευρωνικό δίκτυο

```
final_array = np.array(final_input)
```

Μετατρέπουμε την λίστα με τα δεδομένα εξόδου σε 1d array έτσι ώστε να μπορεί να εισαχθεί στο νευρωνικό δίκτυο

```
final_output = np.array(final_output)
```

Χωρίζουμε το dataset σε training-test με αναλογία 75%-25%

```
X_train, X_test, y_train, y_test = train_test_split(final_array, final_output,
test_size=0.25)
```

Δημιουργούμε το μοντέλο του νευρωνικού δικτύου

```
model = Sequential()
model.add(Dense(12, input_dim=100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Κάνουμε compile το μοντέλο

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Εκπαιδεύουμε το μοντέλο

```
model.fit(X_train, y_train, epochs=20, batch_size=100)
```

Τεστάρουμε το μοντέλο

```
y_pred=model.predict(X_test)
```

Στρογγυλοποιούμε τις τιμές

```
rounded_predictions = [int(round(x[0])) for x in y_pred]
```

Υπολογίζουμε την απόδοση του μοντέλου χρησιμοποιώντας μετρικές f1 score, precision και recall

```
precision = precision_score(y_test, rounded_predictions, average='binary')

print('Precision: %.3f' % precision)

recall = recall_score(y_test, rounded_predictions, average='binary')
print('Recall: %.3f' % recall)

score = f1_score(y_test, rounded_predictions, average='binary')
print('F-Measure: %.3f' % score)
```

Παρακάτω έχουμε τα αποτελέσματα της απόδοσης του μοντέλου

```
Epoch 1/20
12/12 [==============================] - 14s 3ms/step - loss: 0.6545 - accuracy: 0.6746
Epoch 2/20
12/12 [==============================] - 0s 2ms/step - loss: 0.6057 - accuracy: 0.6658
Epoch 3/20
12/12 [==============================] - 0s 2ms/step - loss: 0.5709 - accuracy: 0.6799
Epoch 4/20
12/12 [==============================] - 0s 2ms/step - loss: 0.5432 - accuracy: 0.6962
Epoch 5/20
12/12 [==============================] - 0s 2ms/step - loss: 0.5273 - accuracy: 0.7070
Epoch 6/20
12/12 [==============================] - 0s 2ms/step - loss: 0.5228 - accuracy: 0.7194
Epoch 7/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4973 - accuracy: 0.7468
Epoch 8/20
12/12 [==============================] - 0s 3ms/step - loss: 0.4979 - accuracy: 0.7401
Epoch 9/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4672 - accuracy: 0.7835
Epoch 10/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4675 - accuracy: 0.7701
Epoch 11/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4571 - accuracy: 0.7733
Epoch 12/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4595 - accuracy: 0.7671
```

```
12/12 [==============================] - 0s 2ms/step - loss: 0.4595 - accuracy: 0.7671
Epoch 13/20
12/12 [==============================] - 0s 3ms/step - loss: 0.4377 - accuracy: 0.8032
Epoch 14/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4428 - accuracy: 0.7885
Epoch 15/20
12/12 [==============================] - 0s 3ms/step - loss: 0.4290 - accuracy: 0.8061
Epoch 16/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4464 - accuracy: 0.7967
Epoch 17/20
12/12 [==============================] - 0s 3ms/step - loss: 0.4296 - accuracy: 0.8193
Epoch 18/20
12/12 [==============================] - 0s 3ms/step - loss: 0.4367 - accuracy: 0.8026
Epoch 19/20
12/12 [==============================] - 0s 3ms/step - loss: 0.4130 - accuracy: 0.8234
Epoch 20/20
12/12 [==============================] - 0s 2ms/step - loss: 0.4208 - accuracy: 0.8213
```

```
Precision: 0.779
Recall: 0.552
F-Measure: 0.646
```