# Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

ΑΝΑΦΟΡΑ
Εργαστηριακή Άσκηση
Εαρινό Εξάμηνο 2020-21

Ομάδα:
Φουσέκη Αθηνά, 1059623, up1059623@upnet.gr, 4ο έτος

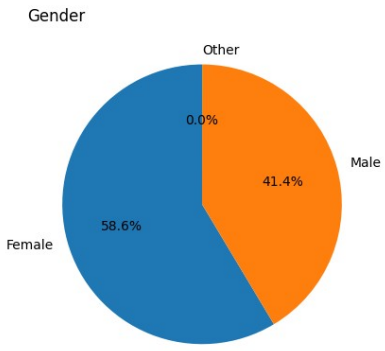
## Ερώτημα 1

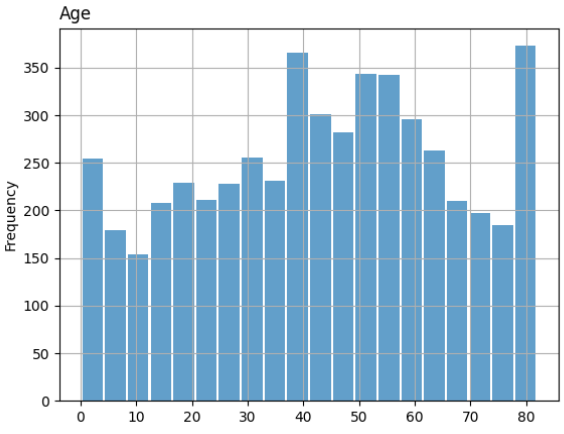
### Α. Να πραγματοποιηθεί ανάλυση του dataset και γραφική αναπαράσταση αυτής.

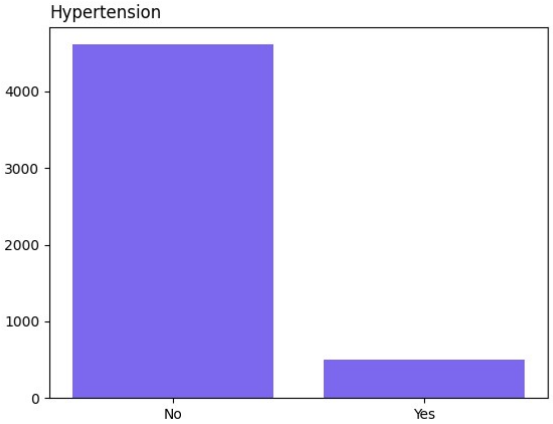
Μας έχει δοθεί το αρχείο healthcare-dataset-stroke-data.csv, το οποίο περιέχει ιατρικά δεδομένα 5110 ασθενών. Οι πληροφορίες αυτές συμπεριλαμβάνουν ένα id, το φύλο, την ηλικία, την αρτηριακή πίεση, εάν κάποιος έχει παντρευτεί, τον τύπο εργασίας, το περιβάλλον κατοικίας, τα επίπεδα γλυκόζης στο αίμα, το bmi, την σχέση του ασθενή με το κάπνισμα και εάν ο ασθενής έχει πάθει ποτέ εγκεφαλικό.
Για να δούμε τις παραπάνω πληροφορίες χρησιμοποιήσαμε την συνάρτηση info(), έχοντας περάσει το dataset σε ένα pandas dataframe.

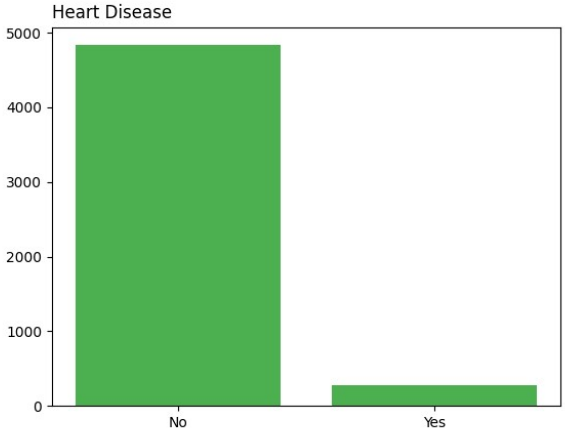
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
None
```

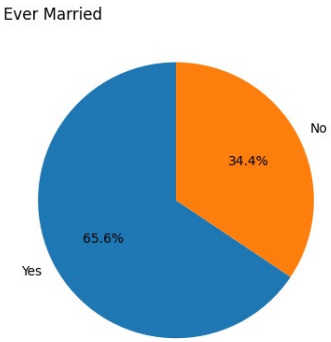
<u>Γραφήματα Στατιστικών του Dataset</u>

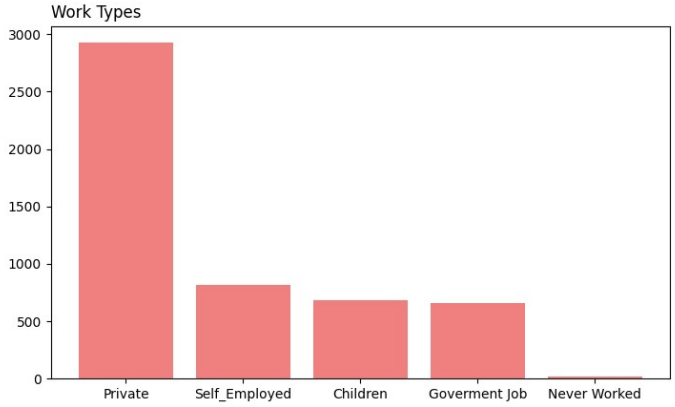


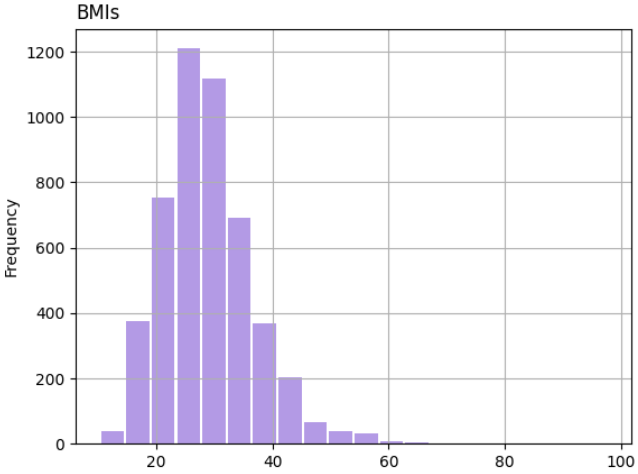



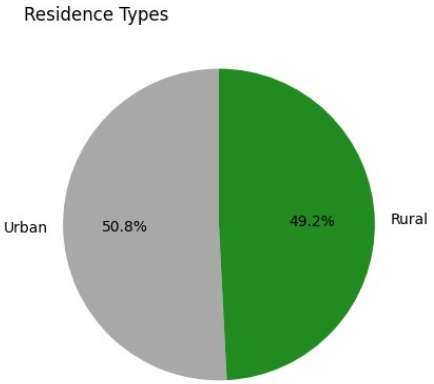



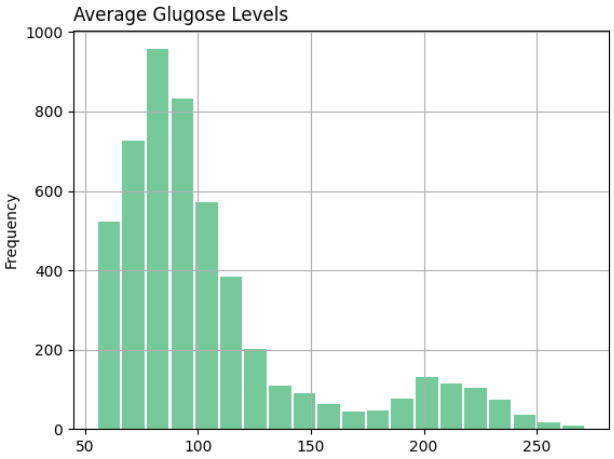



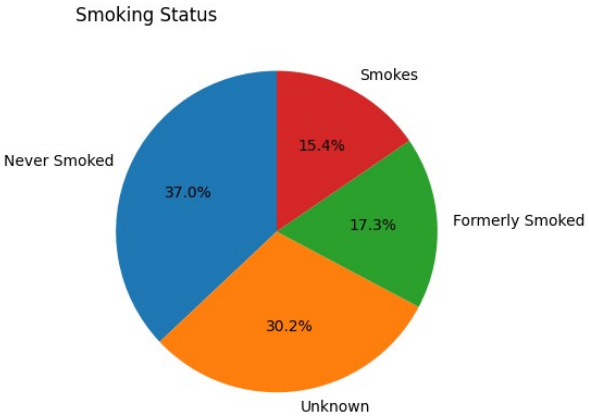

Ever Married
No
34.4%
65.6%
Yes

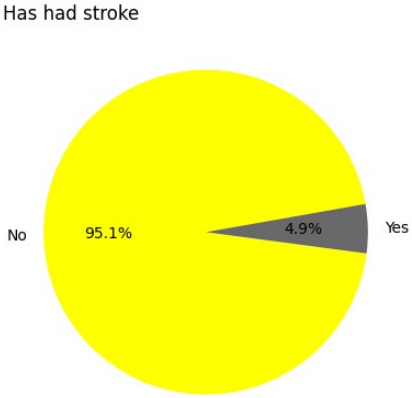

Work Types
3000
2500
2000
1500
1000
500
0
Private
Self_Employed
Children
Goverment Job
Never Worked


BMIs
Frequency
1200
1000
800
600
400
200
0
20
40
60
80
100

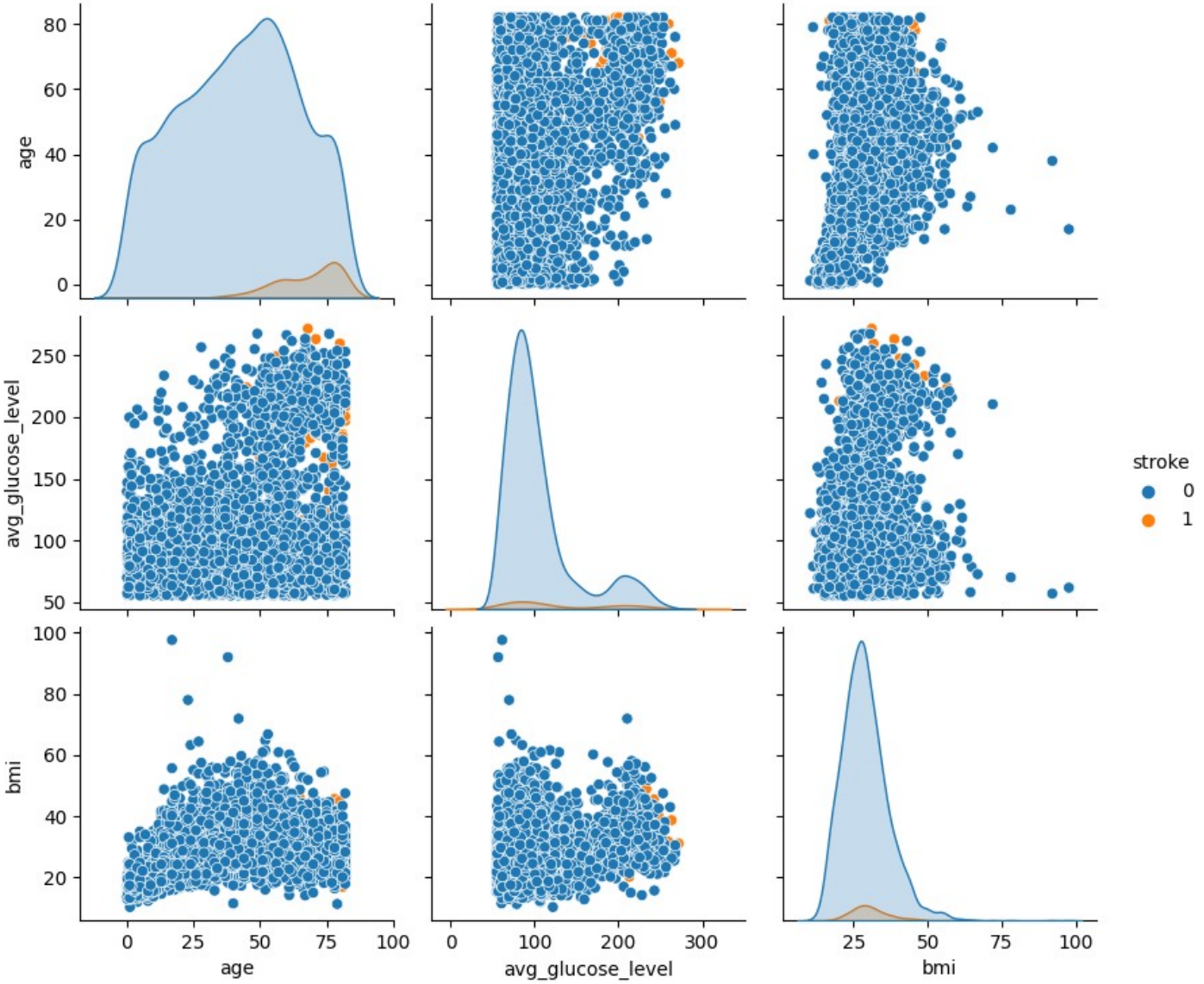

Residence Types
Urban
50.8%
49.2%
Rural

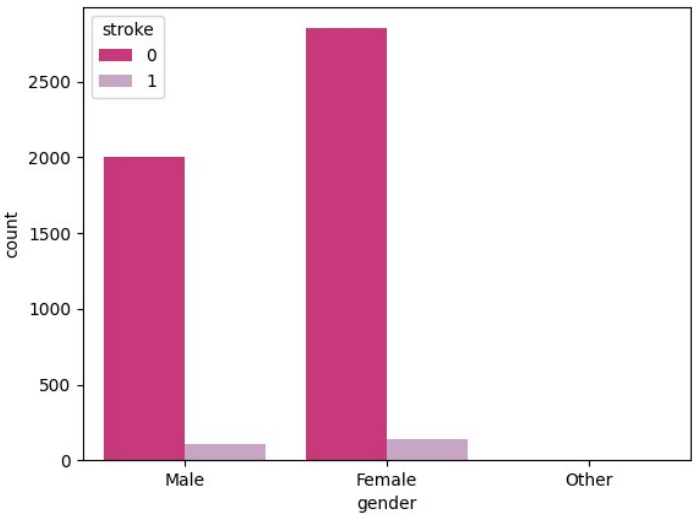

Average Glugose Levels
Frequency
1000
800
600
400
200
0
50
100
150
200
250


Smoking Status
Smokes
15.4%
Never Smoked
37.0%
17.3%
Formerly Smoked
30.2%
Unknown

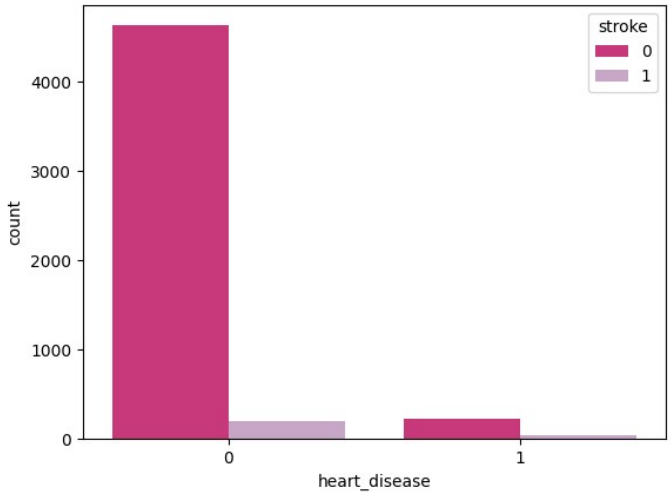

Has had stroke
No
95.1%
4.9%
Yes

Γραφήματα Στατιστικών σε σχέση με το γνώρισμα “stroke”



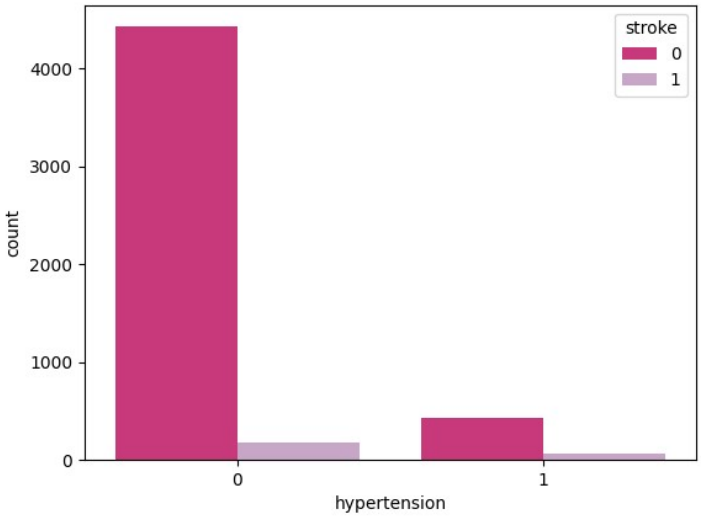




Παρατηρούμε ότι η αναλογία για την εμφάνιση εγκεφαλικού επεισοδίου ανάλογα με το φύλο δεν είναι ιδιαίτερα διαφορετική για άνδρες και γυναίκες.

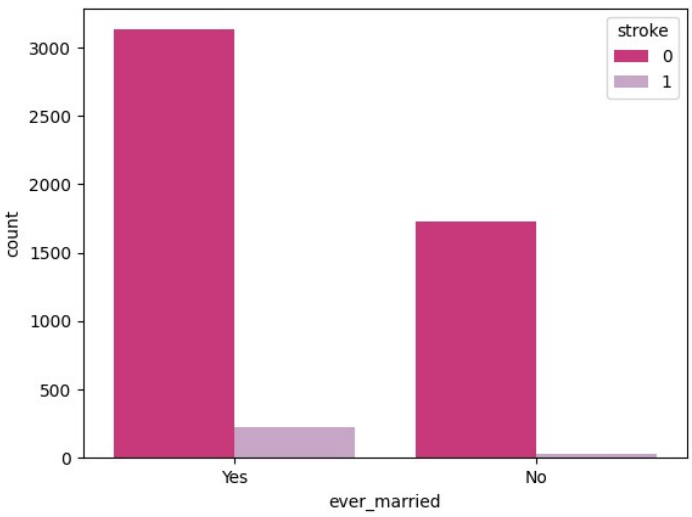



Παρατηρούμε ότι η πιθανότητα να έχει κάποιος εγκεφαλικό επεισόδιο εάν έχει κάποια καρδιοπάθεια, είναι μεγαλύτερη από το να είναι υγιής και να πάθει εγκεφαλικό.

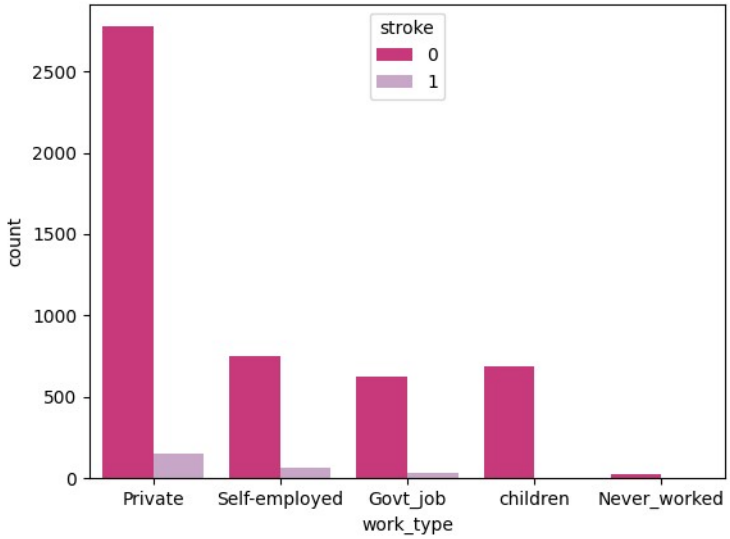
Από όσα άτομα δεν έχουν κάποια καρδιοπάθεια, τα 4630 (περίπου) δεν έχουν και εγκεφαλικό επεισόδιο και μόνο 200 έχουν. Από όσα άτομα έχουν κάποια καρδιοπάθεια, τα 230 δεν έχουν πάθει εγκεφαλικό, ενώ τα 50 έχουν.



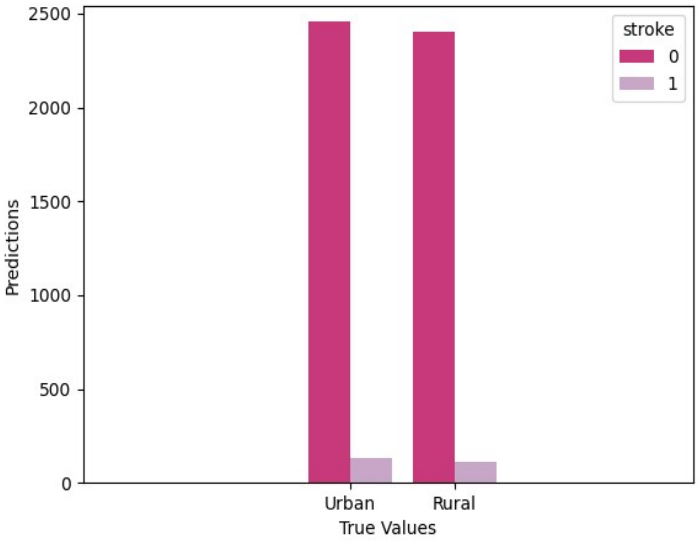

Και σε αυτό το γράφημα, παρατηρούμε ότι είναι πιο πιθανό να προκύψει εγκεφαλικό επεισόδιο σε ασθενή, εάν έχει υψηλή αρτηριακή πίεση.




Εδώ παρατηρούμε ότι η πιθανότητα για εμφάνιση εγκεφαλικού επεισοδίου σε ασθενή είναι περίπου 6,5% στην περίπτωση που έχει παντρευτεί, ενώ μόλις 1,1% στην περίπτωση που δεν έχει παντρευτεί ποτέ.




Στο γράφημα αυτό δεν βλέπουμε κάποια σημαντική διαφοροποίηση στην εμφάνιση εγκεφαλικού επεισοδίου αναφορικά με τους τύπους επαγγέλματος private, self-employed και government job. Σημαντικά χαμηλό ποσοστό παρουσιάζει το επάγγελμα children (stay at home parent ή επάγγελμα που αφορά παιδιά).




Και σε αυτό το γράφημα η συσχέτιση εμφάνισης εγκεφαλικού με τον τύπο κατοικίας μοιάζει να μην υπάρχει.

-Για τα διαγράμματα χρησιμοποιήθηκαν οι βιβλιοθήκες matplotlib και seaborn.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

## B.

*Από τη δημιουργία των γραφημάτων έχουμε δει ότι υπάρχει μονό ένα row με την τιμή “Other” στην στήλη genre,  κι επειδή η ύπαρξη του θα δράσει αρνητικά στο μοντέλο μας, την αφαιρούμε από το dataset.

### 1. Αφαίρεση Στήλης

Έχοντας χρησιμοποιήσει την συνάρτηση info() στο προηγούμενο ερώτημα, μπορούμε να εξάγουμε την απαραίτητη πληροφορία για να εντοπίσουμε τα γνωρίσματα στα οποία μπορεί να υπάρχουν missing values (empty cells, wrong data).

Παρατηρούμε ότι η στήλη bmi έχει 5109-4908= 201 missing values, συνεπώς είναι μία από τις υποψήφιες προς αφαίρεση στήλες.

Επειδή είναι δυνατό να έχουμε missing values που δεν είναι απαραίτητα NaN, έτσι ώστε να τα εντοπίσει η info(), π.χ. στην περίπτωση των κατηγορικών δεδομένων, χρησιμοποιούμε την συνάρτηση unique(), έτσι ώστε να δούμε αν σε κάποια άλλη στήλη υπάρχουν τιμές όπου συμβολίζουν την απουσία πληροφορίας.

Αποτελέσματα:

```
smoking_status
['formerly smoked' 'never smoked' 'smokes' 'Unknown']
```

Παρατηρούμε ότι η στήλη smoking_status περιέχει εγγραφές με την τιμή “Unknown”, και χρησιμοποιώντας την value_counts() σε αυτή, βλέπουμε ότι το σύνολο των “προβληματικών” εγγραφών είναι 1544.

```
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
```

Άρα, η στήλη smoking_status  είναι η δεύτερη υποψήφια προς αφαίρεση.

Αφαιρούμε κάθε στήλη με την αντίστοιχη εντολή:

```
df_noBMI=df.drop(['bmi'], axis = 1)  # Remove column 'bmi'
df_noSmoke=df.drop(['smoking_status'], axis = 1)  # Remove column 'smoking_status'
```

### 2. Συμπληρώστε τις τιμές με το μέσο όρο των στοιχείων της στήλης

Αρχικά, δημιουργούμε ένα νέο dataframe, το dfB, αντίγραφο του αρχικού, και δουλεύουμε σε αυτό για το ερώτημα Β, ώστε να κρατήσουμε το αρχικό df στην αρχική του κατάσταση.
Για την στήλη bmi χρησιμοποιούμε την συνάρτηση mean(), για να υπολογίσουμε τον μέσο όρο των αριθμητικών της τιμών, και την συνάρτηση fillna(), με ορίσματα τον μέσο όρο που υπολογίσαμε και την παράμετρο inplace = True, έτσι ώστε η αλλαγή να γίνει στο ίδιο το dataframe (dfB) που καλεί την συνάρτηση.

Καλώντας πάλι την info(), παρατηρούμε ότι οι missing values στην στήλη bmi έχουν συμπληρωθεί.

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5109 non-null   int64
 1   gender             5109 non-null   object
 2   age                5109 non-null   float64
 3   hypertension       5109 non-null   int64
 4   heart_disease      5109 non-null   int64
 5   ever_married       5109 non-null   object
 6   work_type          5109 non-null   object
 7   Residence_type     5109 non-null   object
 8   avg_glucose_level  5109 non-null   float64
 9   bmi                5109 non-null   float64
 10  smoking_status     5109 non-null   object
 11  stroke             5109 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 518.9+ KB
None
```

Στην περίπτωση της στήλης smoking_status, δεν μπορούμε να πάρουμε τον μέσο όρο των τιμών, καθώς τα δεδομένα είναι κατηγορικά. Αντί αυτού, θα αντικαταστήσουμε τις “Unknown” τιμές με την πιο συχνή τιμή αυτής της στήλης. Χρησιμοποιούμε πάλι την value_counts() σε συνδυασμό με την idxmax(), έτσι ώστε να μας επιστραφεί η τιμή με την υψηλότερη συχνότητα στην smoking_status, για να κάνουμε την αντικατάσταση.

```
['formerly smoked' 'never smoked' 'smokes']
```

## 3. Συμπληρώστε τις τιμές χρησιμοποιώντας Linear Regression

Δημιουργούμε ένα αντίγραφο του αρχικού dataframe, (dfB3) πάνω στο οποίο θα δουλέψουμε για αυτό το ερώτημα. Για το Linear Regression μπορούμε να χρησιμοποιήσουμε μόνο αριθμητικά δεδομένα, συνεπώς, για να διατηρήσουμε την πληροφορία που μας παρέχουν τα κατηγορικά δεδομένα, θα τα μετατρέψουμε σε αριθμητικά. Επειδή το label encoding (ή αλλιώς integer encoding) τείνει να υποθέτει ότι υπάρχουν συσχετίσεις μεταξύ των levels του κατηγορικού γνωρίσματος, βάσει των αριθμητικών τιμών που τους έχουν αποδοθεί, θα χρησιμοποιήσουμε το one hot encoding για όσα γνωρίσματα έχουν περισσότερες από 2 κλάσεις.

Χρησιμοποιούμε την συνάρτηση get_dummies(), της βιβλιοθήκης pandas, για να δημιουργήσουμε τις νέες στήλες που θα αντιστοιχούν σε κάθε level ενός κατηγορικού γνωρίσματος και στη συνέχεια, με την συνάρτηση concat() , τις προσθέτουμε στο dataframe. Τέλος, αφαιρούμε την αρχική στήλη του κατηγορικού γνωρίσματος και μία από τις dummy στήλες, για να αποφύγουμε το “dummy variable trap”. Αποφασίσαμε η έξτρα στήλη που θα αφαιρέσουμε να είναι αυτή με τα λιγότερα ‘1’, καθώς τέτοιες στήλες δεν βοηθούν το μοντέλο μας. Ακολουθούμε αυτή την διαδικασία για την στήλη work_type του dataset, αλλά όχι για το γνώρισμα smoking_status, καθώς περιέχει κενά cells και δεν θα το χρησιμοποιήσουμε για την εκπαίδευση του μοντέλου για τη συμπλήρωση των κενών της bmi.

Το επόμενο βήμα, μετά την κωδικοποίηση των κατηγορικών γνωρισμάτων, είναι ο διαχωρισμός του dataset σε training set και testing set. Αρχικά, αφαιρούμε από το dfB3 όλες τις γραμμές όπου έχουν

NaN τιμές για το γνώρισμα bmi και έχουμε το dfB3_clean_bmi. Τις κρατάμε σε ένα άλλο dataframe, dfB3_null_bmi, γιατί είναι αυτές που θα συμπληρώσουμε με το μοντέλο μας στη συνέχεια. Για τα training και test sets θα χρησιμοποιήσουμε το dfB3_clean_bmi. Από το dfB3_clean_bmi dataframe, ορίζουμε ποιες στήλες θα αντιστοιχούν στις ανεξάρτητες μεταβλητές και ποια θα είναι η target μεταβλητή.

```
x_data = dfB3_clean_bmi[['age', 'hypertension', 'heart_disease',
              'avg_glucose_level', 'stroke', 'gender',
              'ever_married', 'work_Govt_job', 'work_Private',
              'work_Self-employed', 'work_children','Residence_type']]#.values
y_data = dfB3_clean_bmi['bmi']#.values
```

Και ο διαχωρισμός των δεδομένων σε train και test sets:

```
X_train, X_test, y_train, y_test = train_test_split(x_data, y_data ,test_size = 0.25, random_state=111, shuffle=True)
```

Έπειτα, θα κάνουμε fitting του linear regression μοντέλου στα training δεδομένα μας και θα το δοκιμάσουμε στο test set.

```
lm = LinearRegression()
model = lm.fit(X_train, y_train)
predictions = lm.predict(X_test)
```

Κατόπιν, μέσω του μοντέλου που δημιουργήσαμε, υπολογίζουμε τις τιμές που λείπουν από τη στήλη bmi, τις συμπληρώνουμε στο dfB3_null_bmi και το ενώνουμε με το dfB3_clean_bmi στο dfB3_fin_a.

Ακολουθούμε την ίδια διαδικασία και για την συμπλήρωση των “Unknown” τιμών της στήλης smoking_status και αποθηκεύουμε τα τελικά αποτελέσματα στο dfB3_fin_b. Η μόνη διαφοροποίηση είναι ότι το smoking_status γνώρισμα ήταν κατηγορικό, οπότε εφαρμόσαμε σε αυτό Label encoding, για να το φέρουμε σε αριθμητική μορφή. Προτιμήσαμε το label encoding, για να έχουμε μία target variable, και όχι 3.

Label Encoding:

```
le = preprocessing.LabelEncoder()
dfB3_clean_smoke['smoking_status'] = le.fit_transform(dfB3_clean_smoke.smoking_status.values)
```

Τέλος, θα ενώσουμε το dfB3_fin_a με τη στήλη smoking_status του dfB3_fin_b, χρησιμοποιώντας ένα inner join, βάσει του id, και θα αποθηκεύσουμε το αποτέλεσμα στο dataframe dfB3_final.

```
dfB3_final = pd.merge(dfB3_fin_b[['id','smoking_status']], dfB3_fin_a[['id', 'age', 'hypertension',
              'heart_disease', 'avg_glucose_level', 'stroke', 'gender', 'ever_married',
              'work_Govt_job', 'work_Private', 'work_Self-employed', 'work_children',
              'Residence_type','bmi']], on=["id"])
```

## 4. Εφαρμόστε k-Nearest Neighbors για να συμπληρώσετε τις τιμές.

Για αυτό το ερώτημα, η προεπεξεργασία των δεδομένων είναι η ίδια με του προηγούμενου υποερωτήματος.

Για τη δημιουργία των δύο μοντέλων KNN (ένα για την συμπλήρωση των κενών στην στήλη bmi και ένα για την αντικατάσταση των “Unknown” τιμών στην στήλη smoking_status) χρησιμοποιήσαμε από την sklearn.model_selection την GridSearchCV, δίνοντας ως παραμέτρους την KneighborsRegressor() και μία ακόμα παράμετρο, η οποία υποδηλώνει ένα range για το πλήθος των

κοντινότερων γειτόνων που επιθυμούμε, έτσι ώστε να βρεθεί το βέλτιστο K και να έχουμε όσο το δυνατόν καλύτερη κατηγοριοποίηση.

```
parameters = {"n_neighbors": range(1, 50)}
gridsearch = GridSearchCV(KNeighborsRegressor(), parameters)
gridsearch.fit(X_train, y_train) #the training set we created in the last subquestion
```

Έχοντας φτιάξει τα δύο μοντέλα KNN, υπολογίζουμε τα missing values μέσω αυτών και εκτελώντας τις ίδιες ενέργειες σύνδεσης και σύνθεσης dataframes με το προηγούμενο υποερώτημα, καταλήγουμε στο τελικό αποτέλεσμα, στο dfB4_final.

Γ. Για τα νέα μητρώα που προκύπτουν στο υποερώτημα Β, να προβλέψετε αν ένας ασθενής είναι επιρρεπής ή όχι να πάθει εγκεφαλικό χρησιμοποιώντας Random Forest χωρίζοντας το dataset σε training-test με αναλογία 75%-25% και να μετρήσετε την απόδοσή του μοντέλου σας χρησιμοποιώντας τις μετρικές f1 score, precision και recall. Παραθέστε τα ευρήματα σας σχετικά με το πόσο επηρεάστηκε η ποιότητα της κατηγοριοποίησης. Στη συνέχεια, προσπαθήστε να βελτιώσετε τα αποτελέσματά σας πειραματιζόμενοι με τις παραμέτρους εισόδου.

Τα dataframes που προέκυψαν από την προεπεξεργασία του ερωτήματος Β, με τα οποία θα δουλέψουμε σε αυτό το ερώτημα είναι:

i. dfB1
ii. dfB2
iii. dfB3_final
iv. dfB4_final

Για να μπορέσουμε να δώσουμε στην είσοδο του μοντέλου random forest τα dataframes από τα υποερωτήματα Β1 και Β2, είναι απαραίτητο να τα επεξεργαστούμε και να μετατρέψουμε όλα τα attributes σε αριθμητικά, καθώς μέχρι τώρα δεν είχε χρειαστεί.

Χρησιμοποιούμε και πάλι label encoding και one hot encoding για να φέρουμε τα dataframes σε αποδεκτή από το μοντέλο μορφή. Για το dfB1, μετατρέπουμε τις ίδιες στήλες που μετατρέψαμε και στο ερώτημα Β3. Για το dfB2 θα κάνουμε one hot στα ίδια γνωρίσματα με το dfB1 και label encoding στο γνώρισμα smoking_status για να υπάρχει συνέπεια με τα υπόλοιπα dataframes.

Για το μοντέλο random forest χρησιμοποιήσαμε την συνάρτηση RandomForestClassifier από την sklearn.ensemble. Η αρχική παραμετροποίηση που εφαρμόσαμε ήταν να θέσουμε το πλήθος των estimators σε 30 και το criterion ως εντροπία. Επιπλέον, επειδή το dataset που έχουμε είναι unbalanced, αφού στο το target γνώρισμα stroke, το 95,1% των records ανήκουν στην κλάση 0 , ενώ, μόλις το 4,9% στην 1. Κάτι τέτοιο θα ανάγκαζε το μοντέλο μας να προβλέπει πολύ πιο συχνά ότι κάποιος δεν είναι επιρρεπής σε εγκεφαλικό επεισόδιο , με την πρόβλεψη εμφάνισης εγκεφαλικού να είναι πολύ πιο “δύσκολη”. Συνεπώς, για να μπορέσει το μοντέλο μας να διαχειριστεί αυτό το dataset, θα του αποδόσουμε βάρη στις κλάσεις 0 και 1. Το βάρος για κάθε κλάση δίνεται από τον τύπο: $w_j$=num_of_samples/(num_of_classes * num_of_samples$_j$), για την κλάση j.
Θα χρησιμοποιήσουμε τις μετρικές f1 score, precision και recall για να αξιολογήσουμε το μοντέλο μας.

F1 score

Το f1 score μπορούμε να το θεωρήσουμε ως το weighted average των precision και recall. Η συμμετοχή των precision και recall στο f1 score είναι ισότιμη.

Τύπος:

F1 = 2*(precision * recall)/(precision + recall)

Best value: 1

Worst value: 0

Precision

Το precision είναι ο αναλογία tp/(tp +fp), όπου tp είναι το πλήθος των αληθώς θετικών τιμών και fp είναι το πλήθος των ψευδώς θετικών τιμών. Το precision, δηλαδή, αφορά την ικανότητα του κατηγοριοποιητή να μην κατηγοριοποιεί ως θετικό ένα δείγμα που είναι αρνητικό.

Best value: 1

Worst value: 0

Recall

Το recall είναι η αναλογία tp / (tp + fn), όπου tp είναι το πλήθος των αληθώς θετικών και fn το πλήθος των ψευδώς αρνητικών. Το recall είναι δηλαδή η ικανότητα του κατηγοριοποιητή να βρίσκει όλα τα θετικά δείγματα.

Best value: 1

Worst value: 0

Τα πρώτα μας αποτελέσματά ήταν, για 50 estimators:

```
On dfB1
Training Accuracy is:  0.9966066301226834
Testing Accuracy is:  0.9381846635367762
f1 score: 0.5193086866474635
Precision: 0.5676010301109351
Recall: 0.5163627863487611

On dfB2
Training Accuracy is:  0.9973897154789872
Testing Accuracy is:  0.9420970266040689
f1 score: 0.48509266720386784
Precision: 0.47289866457187746
Recall: 0.49793217535153017

On dfB3
Training Accuracy is:  0.9973897154789872
Testing Accuracy is:  0.9444444444444444
f1 score: 0.48571428571428577
Precision: 0.4725920125293657
Recall: 0.4995860927152318

On dfB4
Training Accuracy is:  0.998172800835291
Testing Accuracy is:  0.9436619718309859
f1 score: 0.48550724637681164
Precision: 0.4725705329153605
Recall: 0.4991721854304636
```

Σκεφτήκαμε για κάθε ένα από τα παραπάνω dataframes να τρέξουμε το μοντέλο random forest και να δοκιμάσουμε διάφορες παραμετροποιήσεις.

Για να επηρεάσουμε τις μετρικές θα δοκιμάσουμε να “παίξουμε” και με τις υπόλοιπες παραμέτρους του RandomForestClassifier, όπως max_features (μέγιστος αριθμός features που μπορεί να έχει ο random forest σε ένα δέντρο ) και max_depth (το πιο μακρύ μονοπάτι στο δέντρο, από τη ρίζα στα φύλλα ). Θα επιλέξουμε n_estimators = 50.

Οι μέγιστες τιμές μετρικών για κάθε dataframe είναι:

| | dfB1 | dfB2 | dfB3 | dfB4 |
|---|---|---|---|---|
| F1 score | 0.509 | 0.511 | 0.519 | 0.517 |
| Precision | 0.553 | 0.554 | 0.555 | 0.554 |
| Recall | 0.738 | 0.739 | 0.735 | 0.734 |
| max_features | sqrt | sqrt | log2 | sqrt |
| max_length | 3 | 4 | 2 | 2 |

*Θα συμπεριλάβουμε κι ένα αρχείο txt που θα περιέχει και τις υπόλοιπες τιμές των μετρικών, για άλλες τιμές παραμέτρων.

Παρατηρούμε ότι τα υψηλότερα scores που επιτυγχάνουμε είναι από τα dataframes dfB2 (means) και dfB3 (linear regression).

```
# Tunning Random Forest
from itertools import product
n_estimators = 50
max_features = [1, 'sqrt', 'log2']
max_depths = [None, 2, 3, 4, 5, 6, 7]
for f, d in product(max_features, max_depths): # with product we can iterate through all possible combinations
    rf = RandomForestClassifier(n_estimators=n_estimators,
                                criterion='entropy',
                                max_features=f,
                                max_depth=d,
                                n_jobs=2, class_weight={0: 0.52, 1: 10.5},
                                random_state=1337)
    rf.fit(X_trainB2, y_trainB2)
    prediction_test2 = rf.predict(X=X_testB2)
    prediction_test2 = [round(x) for x in prediction_test2]

    print('Classification accuracy on test set with max features = {} and max_depth = {}: {:.3f}'.format(f, d, accuracy_score(y_testB2,prediction_test2)))
    print("\tf1 score: {}".format(f1_score(y_testB2, prediction_test2, average='macro')))
    print("\tPrecision: {}".format(precision_score(y_testB2, prediction_test2, average='macro', labels=np.unique(prediction_test2))))
    print("\tRecall: {}\n\n".format(recall_score(y_testB2, prediction_test2, average="macro")))
```

Η εφαρμογή μας αφορά τον έλεγχο του κατά πόσο ένας ασθενής με κάποια συγκεκριμένα χαρακτηριστικά είναι επιρρεπής στο να πάθει εγκεφαλικό. Θα προτιμούσαμε δηλαδή να υπολογίζουμε περισσότερα false positives, και να δράσουμε προληπτικά ,ώστε να λειτουργήσουμε προληπτικά για την περίπτωση εγκεφαλικού επεισοδίου, παρά να έχουμε περισσότερα false negatives και να αμεληθούν περιστατικά ασθενών που να έχουν την ανάγκη για συγκεκριμένη φροντίδα. Συνεπώς, η μετρική που θα θέλαμε να μεγιστοποιήσουμε είναι το recall, καθώς μας δείχνει πόσα από τα positives έχουν επιστραφεί και επιλέγουμε ως παραμέτρους max_features=sqrt και max_depth=4, εκπαιδεύοντας με το μοντέλο με το dfB2.

## Οι βιβλιοθήκες που χρησιμοποιήθηκαν και συγκεκριμένες συναρτήσεις:

pandas as pd
matplotlib.pyplot as plt
seaborn as sns
re
numpy as np
sklearn:

- sklearn.model_selection import train_test_split
- sklearn.linear_model import LinearRegression
- sklearn import preprocessing
- sklearn.neighbors import KNeighborsRegressor
- math import sqrt
- sklearn.model_selection import GridSearchCV
- sklearn.tree import DecisionTreeClassifier, export_graphviz
- sklearn.ensemble import RandomForestClassifier
- sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,mean_squared_error