Αναφορά Προτζεκτ Εξόρυξη Δεδομένων και Μηχανική Μάθηση Ακαδημαικό Έτος 2020-2021

Τα μέλη της ομάδας μας είναι τα εξής:

Επίθετο	Όνομα	Α.Μ.	Email	Έτος
Γιαννούλης	Διονύσης	1054301	up1054301@ upnet.gr	5ο
Μορφόπουλος	Μάριος Αλέξανδρος	1058102	up1058102@ upnet.gr	5ο

Ερώτημα 1:

Α.Στο ερώτημα αυτό πραγματοποιήσαμε ανάλυση του dataset που μας δηλαδή το healthcare-dataset-stroke-data.csv . Αρχικά κάναμε μια προεπεξεργασία των δεδομένων μας στην οποία αφαιρέσαμε την στήλη (column id),γεμίσαμε τις NAN values τις "bmi". Και ελέγξαμε τα φύλλα(gender) που μας έχουν δοθεί και αφαιρέσαμε την κατηγορία "other".Στην συνέχεια κάναμε κανονικοποίηση τις αριθμητικές τιμές του dataset και μετά τις κάναμε διακριτές.

Παραθέτουμε τον κώδικα του ερωτήματος Α.

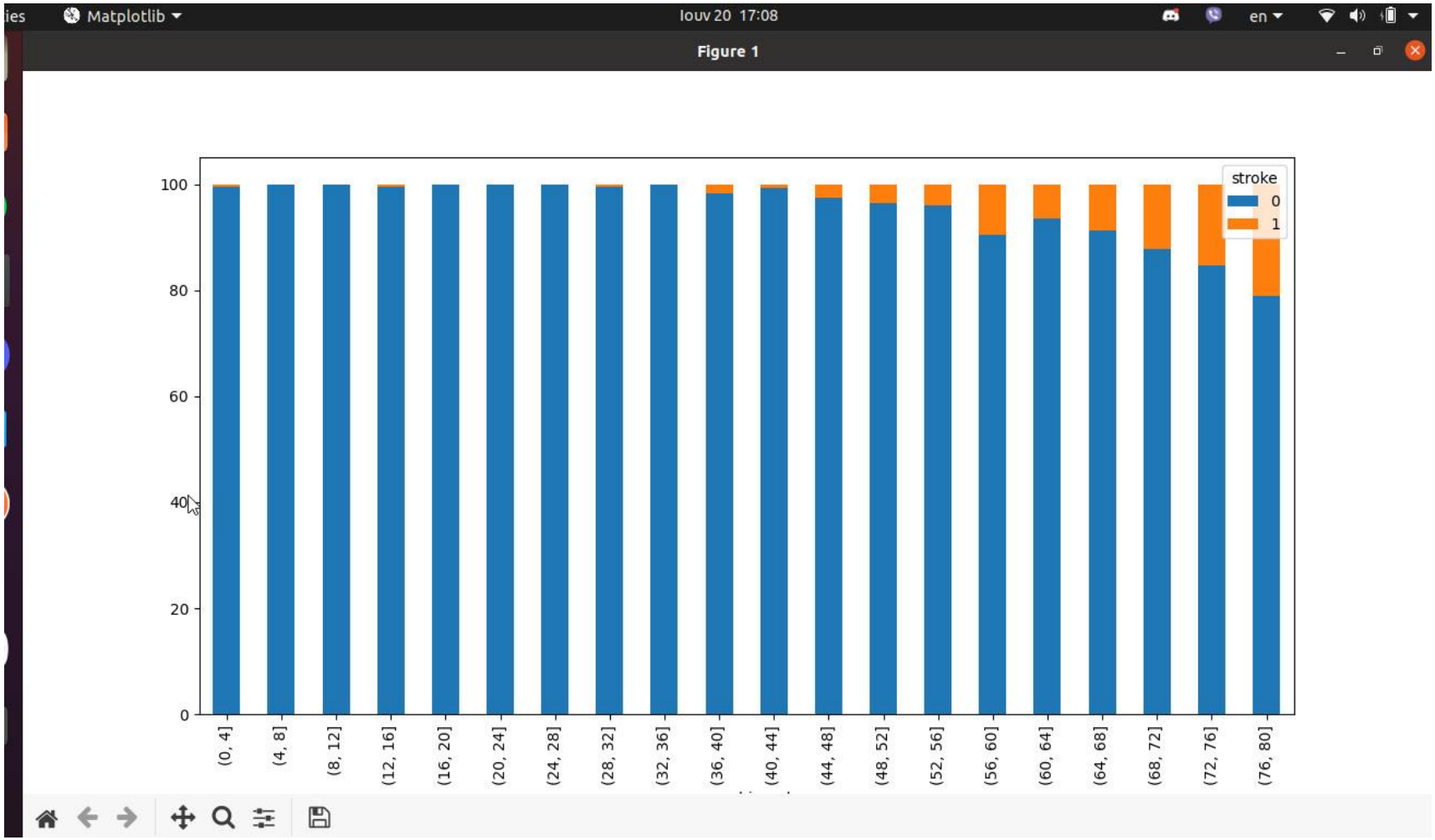
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math

def preprocessing():
    df = pd.read_csv('strokes.csv')
    #afairesi column id
    df.drop(columns=['id'],inplace=True)
    #gemizoume tis NaN values twn bmi
    df['bmi'].fillna(np.round(df['bmi'].mean(), 1), inplace = True)
    #elegxos twn genwn pou mas exoun dwthei
    gen = df['gender'].value_counts()
    #exoume ena other kai to afairoume
    df = df[df['gender'] != 'Other']
    return df

df = preprocessing()

def normalization_columns(df):
```

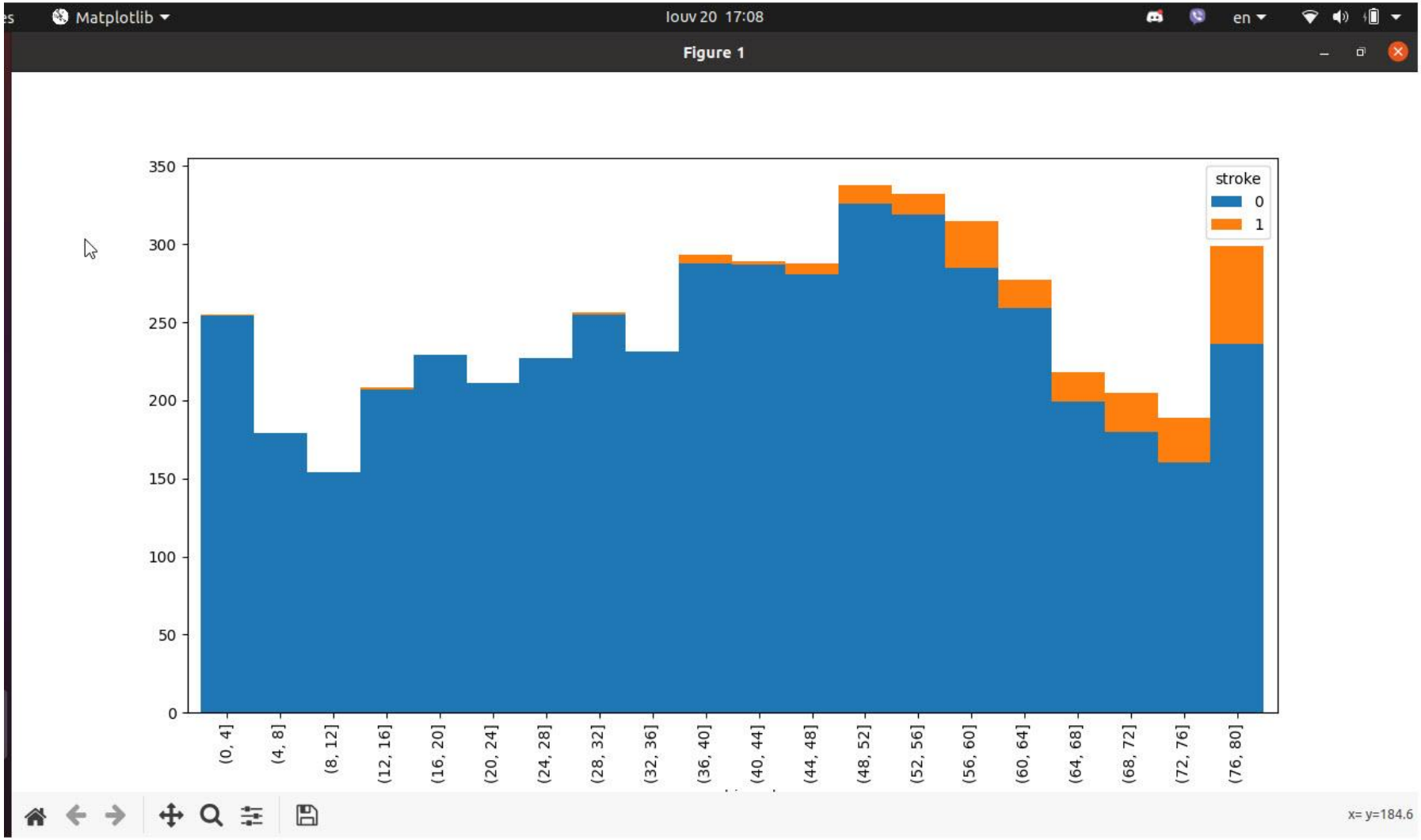
```
    columns_to_norm = ['age','bmi','avg_glucose_level']
    for i in columns_to_norm:
        df[i+'_norm'] = (df[i]-df[i].min())/(df[i].max()-df[i].min())
    return df
df = normalization_columns(df)

def discr_er_range_bin(df):
    columns_to_norm = ['age','bmi','avg_glucose_level']
    for i in columns_to_norm:
        df[i+'_binned'] = pd.cut(df[i], np.arange(int(df[i].min()),int(df[i].max()), int((df[i].max()-df[i].min())/20)))
    return df
df = discr_er_range_bin(df)

def get_stacked_bar_chart(column):
    # Get the count of records by column and stroke
    df_pct = df.groupby([column, 'stroke'])['age'].count()
    # Create proper DataFrame's format
    df_pct = df_pct.unstack()
    return df_pct.plot.bar(stacked=True, figsize=(6,6), width=1)
```

```
def get_100_percent_stacked_bar_chart(column, width = 0.5):
    # Get the count of records by column and stroke
    df_breakdown = df.groupby([column, 'stroke'])['age'].count()
    # Get the count of records by gender
    df_total = df.groupby([column])['age'].count()
    # Get the percentage for 100% stacked bar chart
    df_pct = df_breakdown / df_total * 100
    # Create proper DataFrame's format
    df_pct = df_pct.unstack()
    return df_pct.plot.bar(stacked=True, figsize=(6,6), width=width)

df4 = get_stacked_bar_chart('age_binned')
plt.show()

df5 = get_100_percent_stacked_bar_chart('age_binned', width = 0.5)
plt.show()

fig,axes = plt.subplots(4,2,figsize = (15,15))
fig.suptitle("Count plot for categorical features")
#gender
sns.countplot(ax=axes[0,0],data=df,x='gender')
#smoking_status
sns.countplot(ax=axes[0,1],data=df,x='smoking_status')
```

```
#heart_disease
sns.countplot(ax=axes[1,0],data=df,x='heart_disease')
#ever_married
sns.countplot(ax=axes[1,1],data=df,x='ever_married')
#work_type
sns.countplot(ax=axes[2,0],data=df,x='work_type')
#Residence_type
sns.countplot(ax=axes[2,1],data=df,x='Residence_type')
#hypertension
sns.countplot(ax=axes[3,0],data=df,x='hypertension')
#stroke
sns.countplot(ax=axes[3,1],data=df,x='stroke')

plt.show()
```

Τέλος οι γραφικές αναπαραστάσεις είναι οι ακόλουθες.


Matplotlib
Iouv 20 17:08
en
Figure 1
stroke
0
1
100
80
60
40
20
0
(0, 4]
(4, 8]
(8, 12]
(12, 16]
(16, 20]
(20, 24]
(24, 28]
(28, 32]
(32, 36]
(36, 40]
(40, 44]
(44, 48]
(48, 52]
(52, 56]
(56, 60]
(60, 64]
(64, 68]
(68, 72]
(72, 76]
(76, 80]


Matplotlib
Iouv 20 17:08
en
Figure 1
stroke
0
1
350
300
250
200
150
100
50
0
(0, 4]
(4, 8]
(8, 12]
(12, 16]
(16, 20]
(20, 24]
(24, 28]
(28, 32]
(32, 36]
(36, 40]
(40, 44]
(44, 48]
(48, 52]
(52, 56]
(56, 60]
(60, 64]
(64, 68]
(68, 72]
(72, 76]
(76, 80]
x= y=184.6

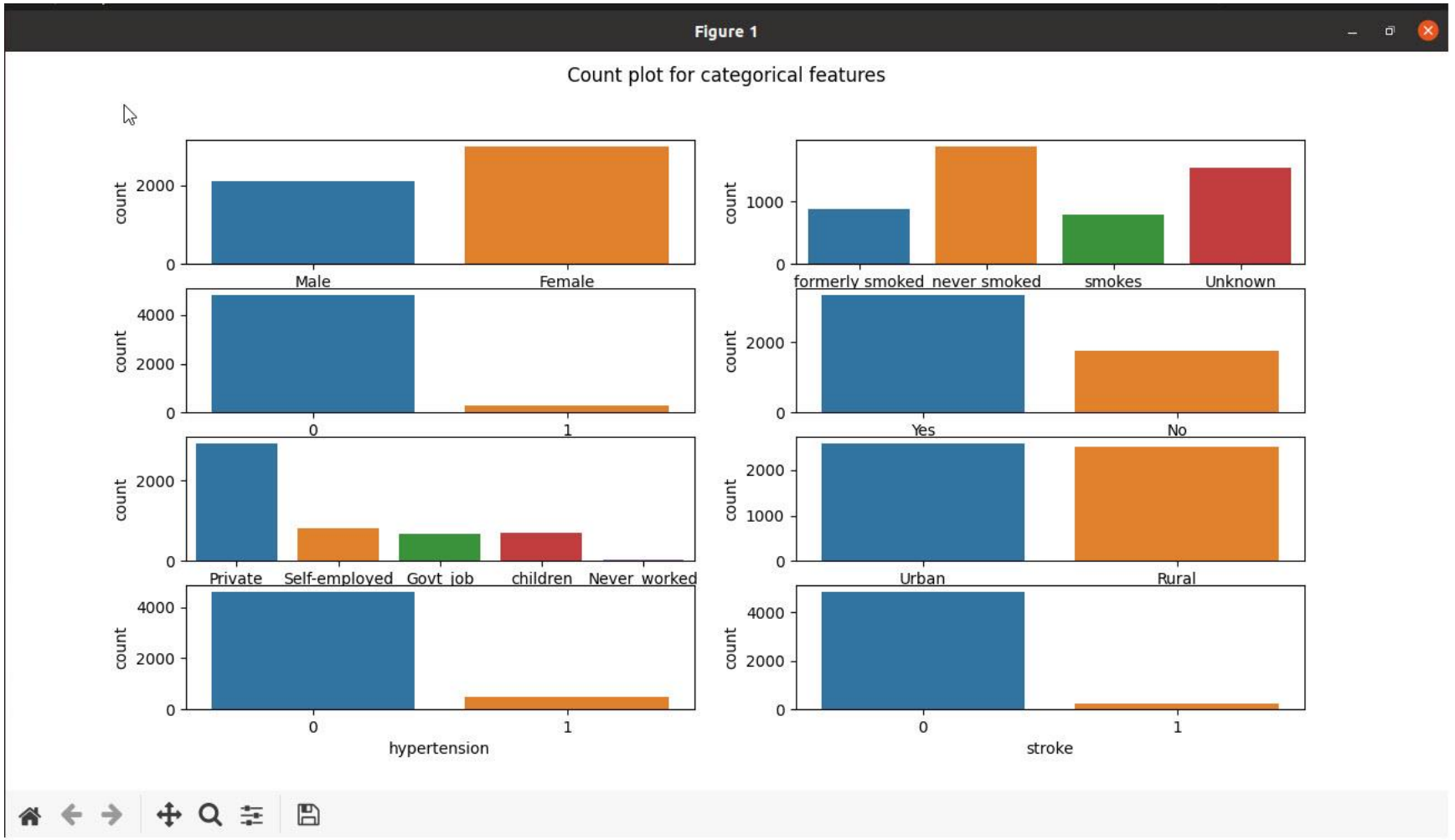


Figure 1
Count plot for categorical features
count
Male
Female
formerly smoked
never smoked
smokes
Unknown
0
1
Yes
No
Private
Self-employed
Govt_job
children
Never_worked
Urban
Rural
hypertension
stroke

Β.Σε αυτό το ερώτημα ορίζουμε 4 συναρτήσεις(delete_nan_values,knn_fill_nan, fill_nan_values_with_mean_of_col,fill_nan_values_with_linear_regr).Οι οποίες απαντάνε στα ζητούμενα ερωτήματα.

Παραθέτουμε τον κώδικα του ερωτήματος Β.

```
import pandas as pd
from fancyimpute import KNN
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

df = pd.read_csv('strokes.csv')
print(df)
df_deleted = df.copy(deep=True)
df_knn = df.copy(deep=True)
df_mean = df.copy(deep=True)
df_linear_regr = df.copy(deep=True)

#function that will delete the rows that have NaN values in our dataframe
def delete_nan_values(df_deleted):
    df_func = df.dropna()
```

```
    return df_func



def knn_fill_nan(df_knn):
    df = df_knn
    # calling the KNN class
    knn_imputer = KNN()
    df1 = df_knn[['age','bmi','avg_glucose_level']]
    # imputing the missing value with knn imputer
    df2 = knn_imputer.fit_transform(df1)
    df3 = pd.DataFrame(df2, columns = ['age','bmi','avg_glucose_level'])
    result = pd.merge(df3, df, how="right", on=['age','bmi','avg_glucose_level'])
    return result



#function that will fill the NaN values of a column  with the mean of values of the column
def fill_nan_values_with_mean_of_col(df_mean):
    #find which columns have NaN values
    col_with_nan = df_mean.columns[df_mean.isna().any()].tolist()
    #each column with NaN values filled with mean
    for i in col_with_nan:
        mean_value = df_mean[i].mean()
        df_mean[i].fillna(value= mean_value,inplace=True)
```

```
    return df_mean
df_fill_nan_with_mean = fill_nan_values_with_mean_of_col(df_mean)
print("Thats a dataframe with the mean values")
print(df_fill_nan_with_mean)

#linear regression
def  fill_nan_values_with_linear_regr(df_linear_regr):
    df_linear_regr['bmi'] = df_linear_regr['bmi'].interpolate(method='linear', limit_direction='both')

deleted_nan_values = delete_nan_values(df_deleted)
print("the dataframe with the deleted nan values")
print(deleted_nan_values)

df_fill_with_linear = fill_nan_values_with_linear_regr(df_linear_regr)
print(df_linear_regr)

df_knn_values = knn_fill_nan(df_knn)
print("the dataframe with the knn values ")
print(df_knn_values)

df_fill_nan_with_mean = fill_nan_values_with_mean_of_col(df_mean)
print("Thats a dataframe with the mean values")
print(df_fill_nan_with_mean)
```

Γ.Εδώ χτίσαμε πάνω στο ερώτημα Β. και προσθέσαμε μια συνάρτηση την RandomForest την οποία την καλούμε για κάθε dataframe που χτίσαμε στο προηγούμενο ερώτημα και υπολογίσαμε και τις μετρικές τους.

Παραθέτουμε τον κώδικα του ερωτήματος Γ.

```
def random_forrest(stroke):
    object_cols = ["gender","ever_married","work_type","Residence_type","smoking_status"]
    label_encoder = LabelEncoder()
    for col in object_cols:
        label_encoder.fit(stroke[col])
        stroke[col] = label_encoder.transform(stroke[col])
    X = stroke.drop('stroke',axis=1)
    y = stroke['stroke']

    #split our data set into train and test
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=100)

    #implement the random forest classifier
    rf = RandomForestClassifier(n_estimators=30, max_depth=10, random_state=1)
    rf.fit(X_train, y_train)
```

```
    y_pred = rf.predict(X_test)

    #result of the predictions
    df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
    recall = metrics.recall_score(y_test, y_pred)
    precision = metrics.precision_score(y_test, y_pred)
    f1_score = metrics.f1_score(y_test, y_pred)
    return df,recall,precision,f1_score



rand_del = random_forrest(deleted_nan_values)
print("Random forrest for the deleted rows")
print(rand_del)
#rand_lin_regr = random_forrest(df_fill_with_linear)
#print("Random forrest for the linear regression",rand_lin_regr)
rand_knn = random_forrest(df_knn_values)
print("Random forrest for the knn")
print(rand_knn)
rand_mean = random_forrest(df_fill_nan_with_mean)
print("Random forrest for the rand_mean")
```

```
print(rand_mean)
```

Ερώτημα 2:Σε αυτό το ερώτημα χωρίζεται στα εξής βασικά σημεία αρχικά επεξεργαζόμαστε λιγο το dataframe μας και κανουμε καποιες τροποποιήσεις.Στην συνέχεια χωρίζουμε το dataframe σε train και test αφου το κανουμε αυτο μετατρέπουμε τα train και test dataframes σε vectors.Τέλος ορίζουμε το ωευρωνικό μας δίκτυο.

Παραθέτουμε τον κώδικα του ερωτήματος 2.

```
import nltk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
from string import punctuation
from nltk.corpus import stopwords
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout

data = pd.read_csv('/home/dionisis/Desktop/exorixi/spam_or_not_spam/spam_or_not_spam.csv')
#make them lowercase
data["email"] = data["email"].str.lower()

X=data['email'].values
y=data['label'].values

#to split our dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

#Vectorization
bow = CountVectorizer()
X_train = bow.fit_transform(X_train)
X_test = bow.transform(X_test)
```

```
#Term Frequency, Inverse Document Frequency
from sklearn.feature_extraction.text import TfidfTransformer
tfidf = TfidfTransformer()
X_train = tfidf.fit_transform(X_train)
X_test = tfidf.transform(X_test)
X_train=X_train.toarray()
X_test=X_test.toarray()

#our neural network
model = Sequential()
model.add(Dense(units=8270,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=4000,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=1000,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=400,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
from tensorflow.keras.callbacks import EarlyStopping
```

```
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)

model.fit(x=X_train,y=y_train,epochs=40,validation_data=(X_test, y_test), verbose=1,callbacks=[early_stop])
```