Μάριος Νεκτάριος Σταματόπουλος

A.M. : 1059383

1η Άσκηση

In [5]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, recall_score, f1_score
df = pd.read_csv('healthcare-dataset-stroke-data\healthcare-dataset-stroke-data.csv'
df.head()
df.drop(index=3116, axis=0, inplace=True)#drop row containing "Other" gender
df.reset_index(drop=True, inplace=True)
```

VISUALISATION

In [20]:
```
from pandas_profiling import ProfileReport
profile = ProfileReport(df, title="Pandas Profiling Report")
profile.to_file("your_report.html")
```

```
Summarize dataset: 100%|██████████| 25/25 [00:13<00:00,  1.90it/s, Completed]
Generate report structure: 100%|██████████| 1/1 [00:06<00:00,  6.55s/it]
Render HTML: 100%|██████████| 1/1 [00:01<00:00,  1.83s/it]
Export report to file: 100%|██████████| 1/1 [00:00<00:00, 45.57it/s]
```

Η ανάλυση και η απεικόνιση του Dataset βρίσκεται στο τέλος

2)Finding missing values

1)

In [6]:
```
#erasing all problematic columns
df_1 = df.drop('bmi', 1,inplace=False)
df_1 = df_1.drop('smoking_status', 1)
df_1.replace('Unknown',np.NaN,inplace=True)
#transformation of data to numerical values
df_1[ ['hypertension', 'heart_disease', 'stroke','ever_married'] ].replace( [0,1] ,[
df_1.Residence_type.replace(['Urban', 'Rural'], [0, 1], inplace=True)
df_1.work_type.replace(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_w
df_1.ever_married.replace(['No','Yes'],[0,1],inplace=True)
df_1.gender.replace(['Male','Female'],[0,1],inplace=True)
df_1.reset_index(drop=True, inplace=True)
```

```
C:\Users\mario\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core
\frame.py:4524: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return super().replace(
```

2)

In [7]:

```
#Finding indices of missing values
df_2=df.copy(deep=True)
missing_indexes_smoking=[(df[col][df[col].eq("Unknown")].index[i], df.columns.get_lo

missing_indexes_bmi=df.index[df['bmi'].isnull() == True]
```

In [8]:

```
#smoking column
missing_indexes_smoking_r=[]


for i in missing_indexes_smoking:
    missing_indexes_smoking_r.append(i[0])

leksiko={}
for i in range(len(df['smoking_status'])):
    if i not in missing_indexes_smoking_r:
        val=df['smoking_status'][i]
        if val not in leksiko:
            leksiko[val]=1
        else:
            leksiko[val]=leksiko[val]+1

print("leksiko:",leksiko)
```

```
leksiko: {'formerly smoked': 884, 'never smoked': 1892, 'smokes': 789}
```

In [9]:

```
#calculating the average of the 3 categories
antistoixia={"never smoked":1,"formerly smoked":2,"smokes":3}
inv_antistoixia = {v: k for k, v in antistoixia.items()}

middle=0
sinolo=0
for key in leksiko:
    middle=middle+antistoixia[key]*leksiko[key]
    sinolo=sinolo+leksiko[key]

middle=middle/sinolo
middle=int(middle+0.5)
print("middle:",middle)
val_to_complete=inv_antistoixia[middle]

for i in missing_indexes_smoking:
    df_2['smoking_status'][i]=val_to_complete

#bmi column
missing_indexes_bmi_r=missing_indexes_bmi
#calculating average
athr,counter=0,0
for i in range(len(df['bmi'])):
    if i not in missing_indexes_bmi_r:
        athr=athr+df['bmi'][i]
        counter=counter+1

average=athr/counter

#filling missing values with average
for i in missing_indexes_bmi_r:
    df_2['bmi'][i]=average

df_2.replace('Unknown',np.NaN,inplace=True)
df_2[ ['hypertension', 'heart_disease', 'stroke','ever_married'] ].replace( [0,1] ,[
df_2.smoking_status.replace(['never smoked', 'formerly smoked', 'smokes'], [0, 1, 2]
```

```
df_2.Residence_type.replace(['Urban', 'Rural'], [0, 1], inplace=True)
df_2.work_type.replace(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_w
df_2.ever_married.replace(['No','Yes'],[0,1],inplace=True)
df_2.gender.replace(['Male','Female'],[0,1],inplace=True)
df_2.reset_index(drop=True, inplace=True)
#df_2[5:10]
```

```
middle: 2
<ipython-input-9-be24a0625680>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_2['smoking_status'][i]=val_to_complete
C:\Users\mario\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core
\indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
<ipython-input-9-be24a0625680>:32: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_2['bmi'][i]=average
C:\Users\mario\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core
\frame.py:4524: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return super().replace(
```

3)

In [10]:

```
df_3=df.copy(deep=True)
#fixing data
df_3.replace('Unknown',np.NaN,inplace=True)
df_3[ ['hypertension', 'heart_disease', 'stroke','ever_married'] ].replace( [0,1] ,[
df_3.smoking_status.replace(['never smoked', 'formerly smoked', 'smokes'], [0, 1, 2]
df_3.Residence_type.replace(['Urban', 'Rural'], [0, 1], inplace=True)
df_3.work_type.replace(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_w
df_3.ever_married.replace(['No','Yes'],[0,1],inplace=True)
df_3.gender.replace(['Male','Female'],[0,1],inplace=True)
df_3.reset_index(drop=True, inplace=True)
```

In [11]:

```
train_data_smoking=[i for i in range(len(df_3)) if i not in missing_indexes_smoking_
train_data_bmi=[i for i in range(len(df_3)) if i not in missing_indexes_bmi_r]
```

In [12]:

```
def intersection (lst1, lst2):
    return list(set(lst1).intersection( set(lst2)) )

#get intersection in order to rfemove all missing values
train_data_indices=intersection(train_data_smoking, train_data_bmi)
```

In [15]:

```
def fillMissing(colName,train_indices,missing_indices):
    global df_3
    df_train=df_3.iloc[train_data_indices]
    Y=df_train[colName]
    X=df_train.drop(columns=['smoking_status','bmi','id'])
```

```
    X= X.astype(float)

    linear_regressor = LinearRegression()  # create object for the class
    linear_regressor.fit(X, Y)  # perform linear regression

    df_predict=df_3.iloc[missing_indices]
    X_predict =df_predict.drop(columns=['smoking_status','bmi','id'])
    X_predict = X_predict.astype(float)
    predictions=linear_regressor.predict(X_predict)

    for i,val in enumerate(predictions):
        val=round(val)
        index=missing_indices[i]
        df_3[colName][index]=val

fillMissing('smoking_status',train_data_indices,missing_indexes_smoking_r)
fillMissing('bmi',train_data_indices,missing_indexes_bmi_r)
```

```
<ipython-input-15-7add478d7937>:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_3[colName][index]=val
<ipython-input-15-7add478d7937>:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_3[colName][index]=val
```

4)

In [16]:

```
df_4=df.copy(deep=True)
#fixing data
df_4.replace('Unknown',np.NaN,inplace=True)
df_4[ ['hypertension', 'heart_disease', 'stroke','ever_married'] ].replace( [0,1] ,[
df_4.smoking_status.replace(['never smoked', 'formerly smoked', 'smokes'], [0, 1, 2]
df_4.Residence_type.replace(['Urban', 'Rural'], [0, 1], inplace=True)
df_4.work_type.replace(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_w
df_4.ever_married.replace(['No','Yes'],[0,1],inplace=True)
df_4.gender.replace(['Male','Female'],[0,1],inplace=True)
df_4.reset_index(drop=True, inplace=True)
```

```
C:\Users\mario\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core
\frame.py:4524: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return super().replace(
```

In [17]:

```
train_cols=set(df_4.columns)

train_cols=list( train_cols-set(["smoking_status",'bmi']) )
train_data = df_4[df_4['smoking_status'].notna() & df_4['bmi'].notna()]
X=train_data[train_cols]
Y=train_data[["smoking_status",'bmi']]
model = KNeighborsRegressor(n_neighbors=25)
model.fit(X,Y)
predictions=model.predict(df_4[train_cols])
for i,val in enumerate(predictions[:,0]):
    predictions[i,0]=round(val)
```

```
for index in missing_indexes_smoking_r:
    df_4['smoking_status'][index]=predictions[index,0]

for index in missing_indexes_bmi_r:
    df_4['bmi'][index]=predictions[index,1]
```

```
<ipython-input-17-91291161af0a>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_4['smoking_status'][index]=predictions[index,0]
<ipython-input-17-91291161af0a>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_4['bmi'][index]=predictions[index,1]
```

2)Random Forest Classifier

In [18]:

```
def RandomForest(df):
    X_cols=set(df.columns)
    X_cols=list( set(train_cols)-set(['stroke']) ) #remove class column from triaini
    X=df[X_cols]
    Y=df['stroke']
    X_train,X_test,Y_train,Y_test=train_test_split(X, Y, test_size=0.25)

    model = RandomForestClassifier(n_estimators=250, bootstrap=False, warm_start=Tru
    model.fit(X_train,Y_train)
    Y_pred = model.predict(X_test)

    print('F1',f1_score(Y_test, Y_pred, average='macro')*100,'%')
    print('Precision',precision_score(Y_test, Y_pred, average='macro')*100,'%' )
    print('Recall',recall_score(Y_test, Y_pred, average='macro')*100,'%')

dataframes=[df_1,df_2,df_3,df_4]
for i,df in enumerate(dataframes):
    print("==============================================================")
    print("Dataframe with method",i,":")
    RandomForest(df)
```

```
==============================================================
Dataframe with method 0 :
F1 48.44695441710367 %
Precision 47.4703557312253 %
Recall 49.46457990115321 %
==============================================================
Dataframe with method 1 :
F1 48.59211584875302 %
Precision 47.446975648075416 %
Recall 49.793899422918386 %
==============================================================
Dataframe with method 2 :
F1 51.591755388891805 %
Precision 56.88096433952787 %
Recall 51.35528547201809 %
==============================================================
Dataframe with method 3 :
F1 51.01771743074258 %
Precision 58.43183609141056 %
Recall 51.13883632923368 %
```

2η Άσκηση

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import pickle
from tensorflow.keras import Sequential
from tensorflow.keras.layers import *
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score
```

In [2]:
```
df = pd.read_csv("spam_or_not_spam\spam_or_not_spam.csv")
df.fillna('', inplace=True)
```

Creation of dictionary for words in Glove Model and save it locally

In [12]:
```
leks={}
with open(f'glove.6B.100d.txt', 'r', encoding='utf-8') as file:
    for l in file.readlines():
        data=l.split()
        word=data[0]
        data=data[1:]
        data=[float(i) for i in data]
        if word not in leks:
            leks[word]=data
```

In [4]:
```
with open('dictionary.pickle', 'wb') as handle:
    pickle.dump(leks, handle, protocol=pickle.HIGHEST_PROTOCOL)

#with open('dictionary.pickle', 'rb') as handle:
#     b = pickle.load(handle)
```

In [6]:
```
leks=b
```

Stemming and Removing stop words

In [13]:
```
from tqdm import tqdm
import gensim
from gensim.parsing.preprocessing import remove_stopwords,stem_text

WORD_LIMIT=100
df_test=df.copy(deep=True)

for ind,email in enumerate (df.email):
    email=remove_stopwords(email)
    email=gensim.utils.simple_preprocess (email)

    email_word_emb=[]
    words_added=0
    for word in email:
        if (words_added>=WORD_LIMIT):
            break

        if word in leks:
            email_word_emb.append(leks[word])
            words_added=words_added+1
    #zero padding
    dx=WORD_LIMIT-words_added
    if (dx>0):
        for i in range(dx):
```

```
            email_word_emb.append( np.zeros(WORD_LIMIT) )
    df_test.email[ind]= np.array( email_word_emb )
```

```
<ipython-input-13-d5e56641d245>:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_test.email[ind]= np.array( email_word_emb )
```

Neural Net creation

In [14]:

```
model = Sequential(name="SpamNet")
model.add(LSTM(int(WORD_LIMIT/2), name='LSTM', input_shape=(WORD_LIMIT, WORD_LIMIT))
model.add(Dense(32, name='Dense_1', activation='relu'))
model.add(Dropout(0.2, name='Dropout_1'))
model.add(Dense(16, name='Dense_2', activation='relu'))
model.add(Dropout(0.05, name='Dropout_2'))
model.add(Dense(1, name='Output', activation="sigmoid"))

model.compile(optimizer='adam', loss='binary_crossentropy',metrics=["binary_accuracy
```

Split data for training and testing

In [15]:

```
X = np.asarray(list(df_test.email)).astype('float32')
Y = np.array(df_test.label)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.25)
```

In [16]:

```
model.summary()
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), batch_size=64, epochs=
```

```
Model: "SpamNet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
LSTM (LSTM)                  (None, 50)                30200
_________________________________________________________________
Dense_1 (Dense)              (None, 32)                1632
_________________________________________________________________
Dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
Dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
Dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
Output (Dense)               (None, 1)                 17
=================================================================
Total params: 32,377
Trainable params: 32,377
Non-trainable params: 0
_________________________________________________________________
```

Out[16]:

```
<tensorflow.python.keras.callbacks.History at 0x1752a4ab0d0>
```

In [17]:

```
Y_pred =[int(i) for i in  model.predict(X_test) + .5]

print(f"F1 score: {f1_score(Y_test, Y_pred, average='macro') *100:.2f}%")
print(f"Precision score: {precision_score(Y_test, Y_pred, average='macro')*100:.2f}%
print(f"Recall Score: {recall_score(Y_test, Y_pred, average='macro')*100:.2f}%")
```

```
F1 score: 94.06%
Precision score: 94.85%
Recall Score: 93.37%
```

Ανάλυση του **healthcare-dataset-stroke** και γραφική αναπαράσταση του

Dataset statistics

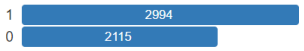
Number of variables	12
Number of observations	5109
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	479.1 KiB
Average record size in memory	96.0 B

Variable types

Numeric	4
Categorical	8

gender
Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB



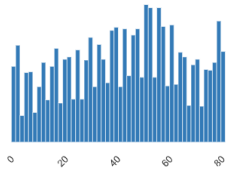
Toggle details

age
Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	104
Distinct (%)	2.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	43.2299863

Minimum	0.08
Maximum	82
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	40.0 KiB



Toggle details

hypertension
Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB

Toggle details

heart_disease
Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB

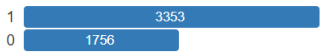
Toggle details

ever_married

Categorical

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB



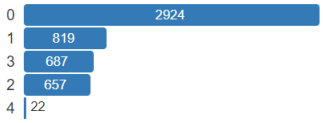
Toggle details

work_type

Categorical

HIGH CORRELATION
HIGH CORRELATION

Distinct	5
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB

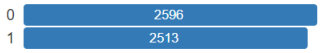


Toggle details

Residence_type

Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB



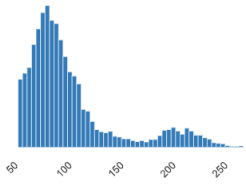
Toggle details

avg_glucose_level

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	3978
Distinct (%)	77.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	106.1403993

Minimum	55.12
Maximum	271.74
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	40.0 KiB



Toggle details

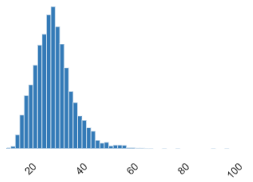
bmi

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	601
Distinct (%)	11.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	28.96097945

Minimum	10.3
Maximum	97.6
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	40.0 KiB

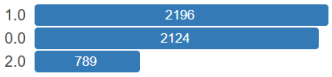


Toggle details

smoking_status

Categorical

Distinct	3
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB



Toggle details

stroke

Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	40.0 KiB

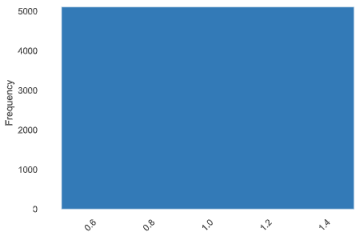
Toggle details

Overview | Categories | Words | Characters

Common Values

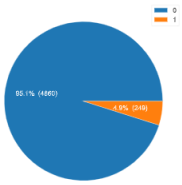
Value	Count	Frequency (%)
0	4860	95.1%
1	249	4.9%

Length



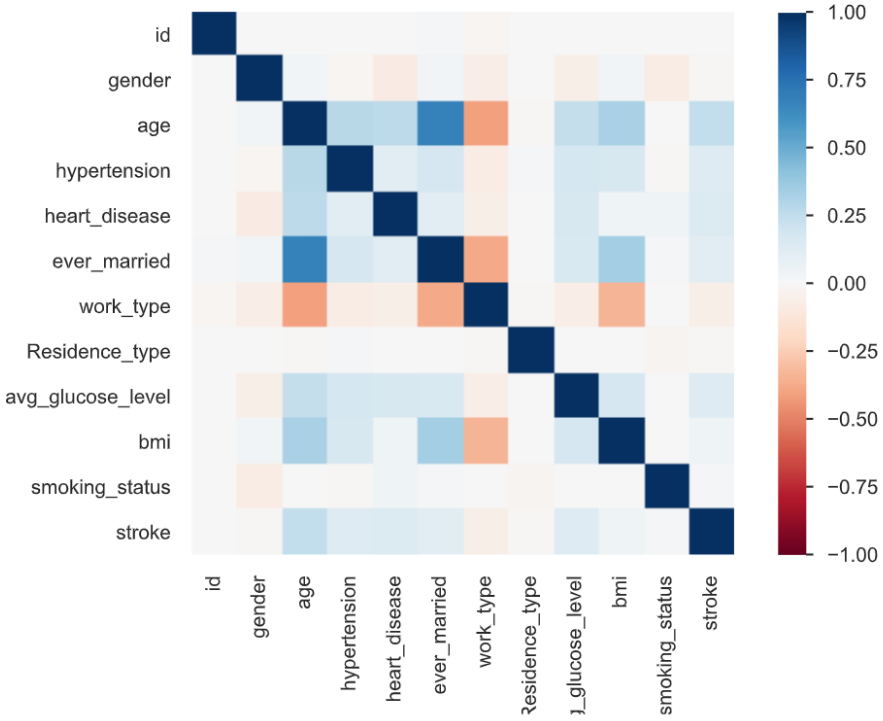
Histogram of lengths of the category

Pie chart

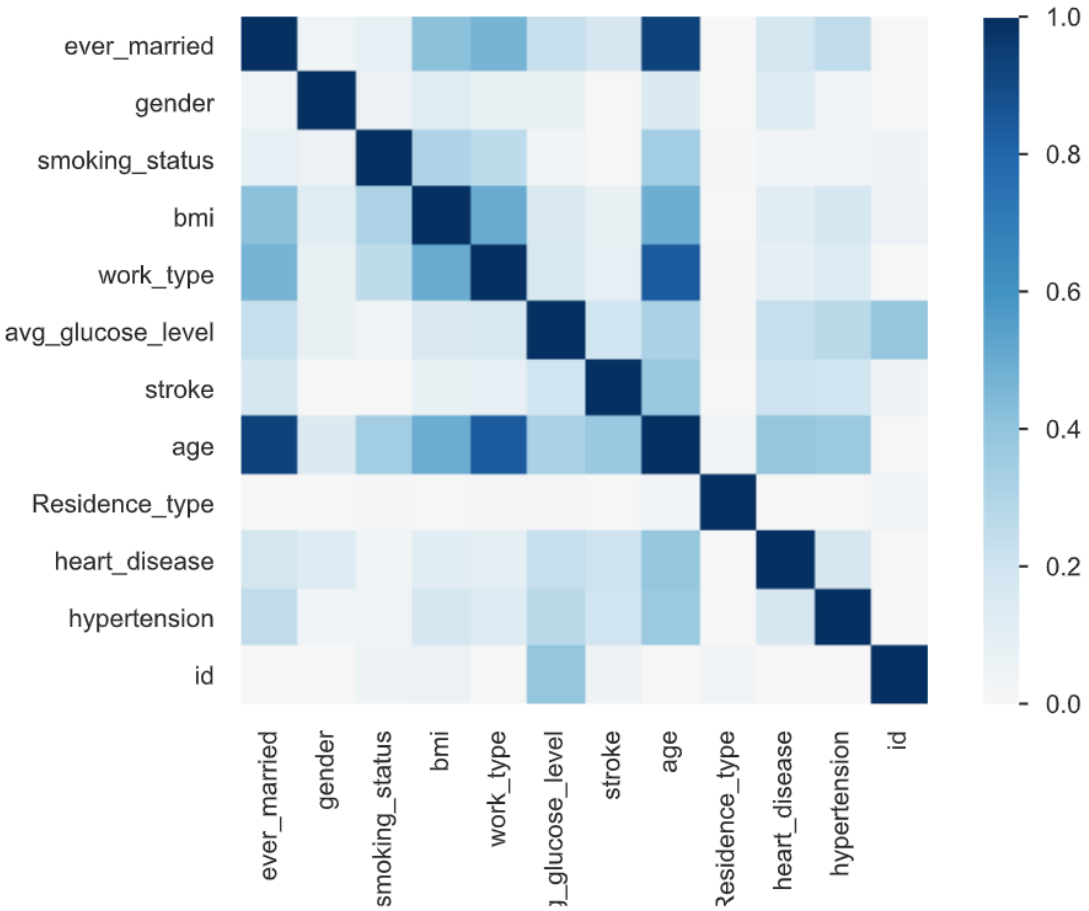


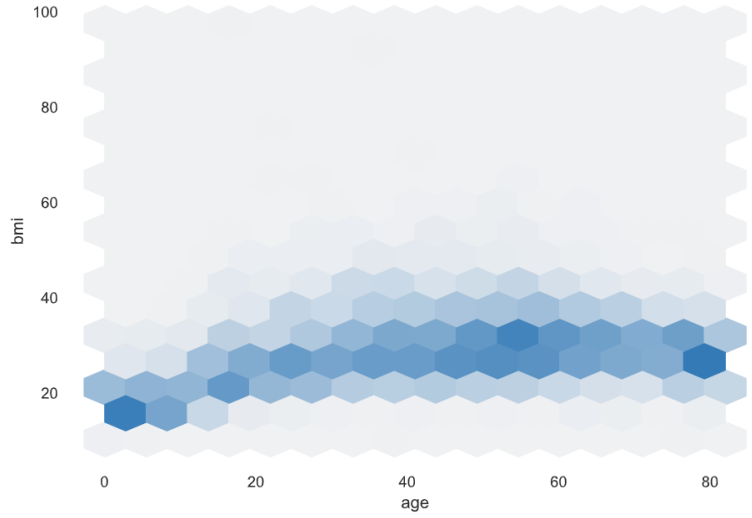
Coorelations

Pearson's



Φκ Coorelation


ever_married
gender
smoking_status
bmi
work_type
avg_glucose_level
stroke
age
Residence_type
heart_disease
hypertension
id
1.0
0.8
0.6
0.4
0.2
0.0


100
80
60
40
20
bmi
0
20
40
60
80
age

`age` is highly correlated with `ever_married`

`ever_married` is highly correlated with `age`

`age` is highly correlated with `ever_married`

`ever_married` is highly correlated with `age`

`age` is highly correlated with `ever_married`

`ever_married` is highly correlated with `age`

`ever_married` is highly correlated with `age`

`bmi` is highly correlated with `work_type`

`work_type` is highly correlated with `bmi` and 1 other fields

`age` is highly correlated with `ever_married` and 1 other fields

`ever_married` is highly correlated with `work_type`

`work_type` is highly correlated with `ever_married`