Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案

Demo 11 – Heart Disease

Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Scenario
- Data Import
- Categorical Data handling
- Correlation heatmap
- Logistic Function
- Machine Learning

Scenario

In Hong Kong, cause of death by heart diseases ranks top 3 for many decades. The situation is similar in other developed countries.

A dataset conducted by US CDC shows that About half of all Americans (47%) have at least 1 of 3 key risk factors for heart disease: high blood pressure, high cholesterol, and smoking. Other key indicator include diabetic status, obesity (high BMI), not getting enough physical activity or drinking too much alcohol.

Data import

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib as mpl
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 import seaborn as sns
7 import missingno as msno
8 from scipy import stats
```

```
1 train = pd.read_csv('heart_2020_cleaned.csv')
2 train.sample(3)
```

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth
30520	No	33.89	Yes	No	No	0.0	0.0
33088	No	32.78	No	No	No	0.0	0.0
88284	No	22.32	No	No	No	0.0	0.0

Overview all columns

```
1 train.describe(include='all')
```

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	Ph
count	319795	319795.000000	319795	319795	319795	319795.00000	319795.000000	319795	319795	319795	319795	319795	
unique	2	NaN	2	2	2	NaN	NaN	2	2	13	6	4	
top	No	NaN	No	No	No	NaN	NaN	No	Female	65-69	White	No	
freq	292422	NaN	187887	298018	307726	NaN	NaN	275385	167805	34151	245212	269653	
mean	NaN	28.325399	NaN	NaN	NaN	3.37171	3.898366	NaN	NaN	NaN	NaN	NaN	
std	NaN	6.356100	NaN	NaN	NaN	7.95085	7.955235	NaN	NaN	NaN	NaN	NaN	
min	NaN	12.020000	NaN	NaN	NaN	0.00000	0.000000	NaN	NaN	NaN	NaN	NaN	
25%	NaN	24.030000	NaN	NaN	NaN	0.00000	0.000000	NaN	NaN	NaN	NaN	NaN	
50%	NaN	27.340000	NaN	NaN	NaN	0.00000	0.000000	NaN	NaN	NaN	NaN	NaN	
75%	NaN	31.420000	NaN	NaN	NaN	2.00000	3.000000	NaN	NaN	NaN	NaN	NaN	
max	NaN	94.850000	NaN	NaN	NaN	30.00000	30.000000	NaN	NaN	NaN	NaN	NaN	

Numerical data

```
1 train.describe().style.background_gradient(cmap = "Set3")
```

	BMI	PhysicalHealth	MentalHealth	SleepTime
count	319795.000000	319795.000000	319795.000000	319795.000000
mean	28.325399	3.371710	3.898366	7.097075
std	6.356100	7.950850	7.955235	1.436007
min	12.020000	0.000000	0.000000	1.000000
25%	24.030000	0.000000	0.000000	6.000000
50%	27.340000	0.000000	0.000000	7.000000
75%	31.420000	2.000000	3.000000	8.000000
max	94.850000	30.000000	30.000000	24.000000

Numerical data definition

- BMI = Body Mass Index (BMI)
- MentalHealth = Thinking about your mental health, for how many days during the past 30 days was your mental health not good?
- PhysicalHealth = Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30
- SleepTime = On average, how many hours of sleep do you get in a 24-hour period?

Categorical data definition

- HeartDisease =Respondents that have ever reported having coronary heart disease (CHD) or myocardial infarction (MI)
- Smoking = Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes]
- AlcoholDrinking = Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week
- DiffWalking = Do you have serious difficulty walking or climbing stairs?
- Sex = Are you male or female?
- Stroke = (Ever told) (you had) a stroke?

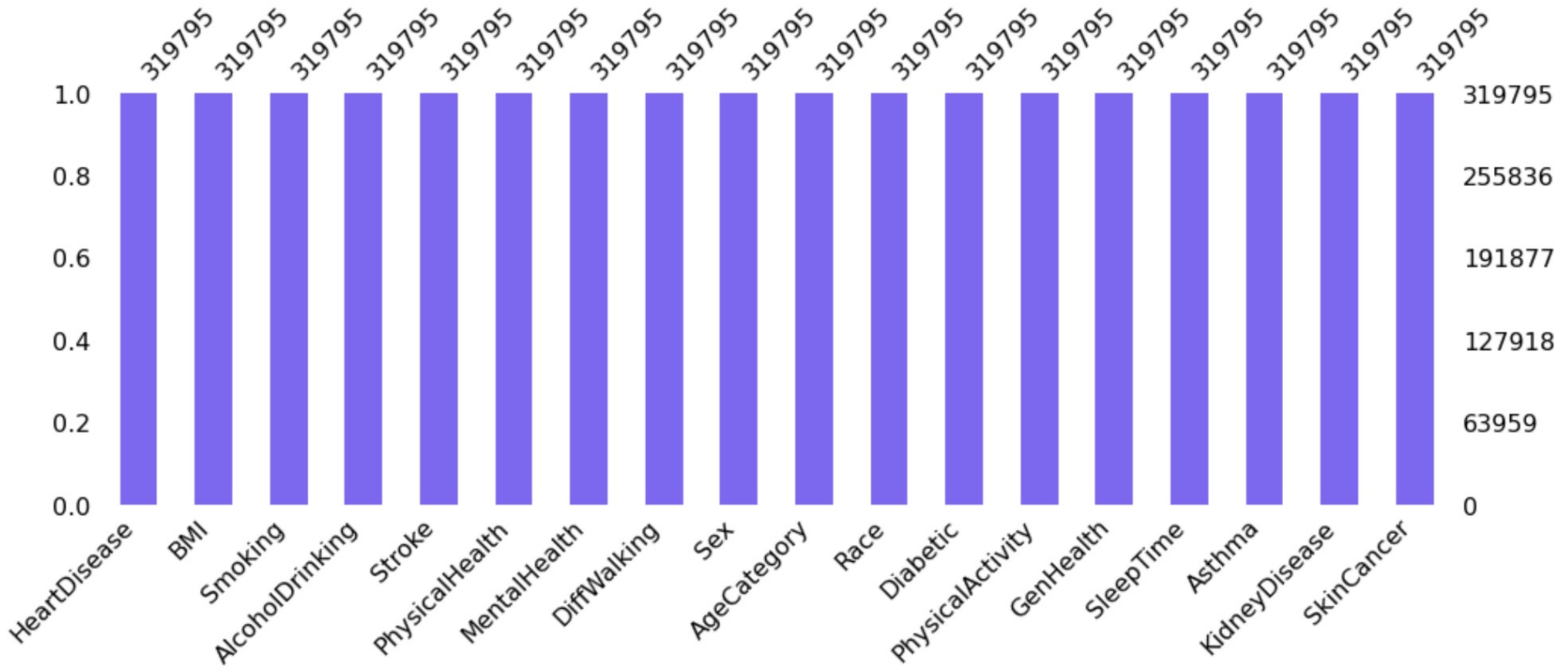
Categorical data definition

- AgeCategory = ['18-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54', '55-59', '60-64', '65-69', '70-74', '75-79', '80 or older']
- Race = Imputed race/ethnicity value
- Diabetic = (Ever told) (you had) diabetes?
- PhysicalActivity = Adults who reported doing physical activity or exercise during the past 30 days other than their regular job
- GenHealth = Would you say that in general your health is...
- PhysicalHealth = Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30
- KidneyDisease = Not including kidney stones, bladder infection or incontinence, were you ever told you had kidney disease?
- SkinCancer=(Ever told) (you had) skin cancer?

Check missing data

All Clear

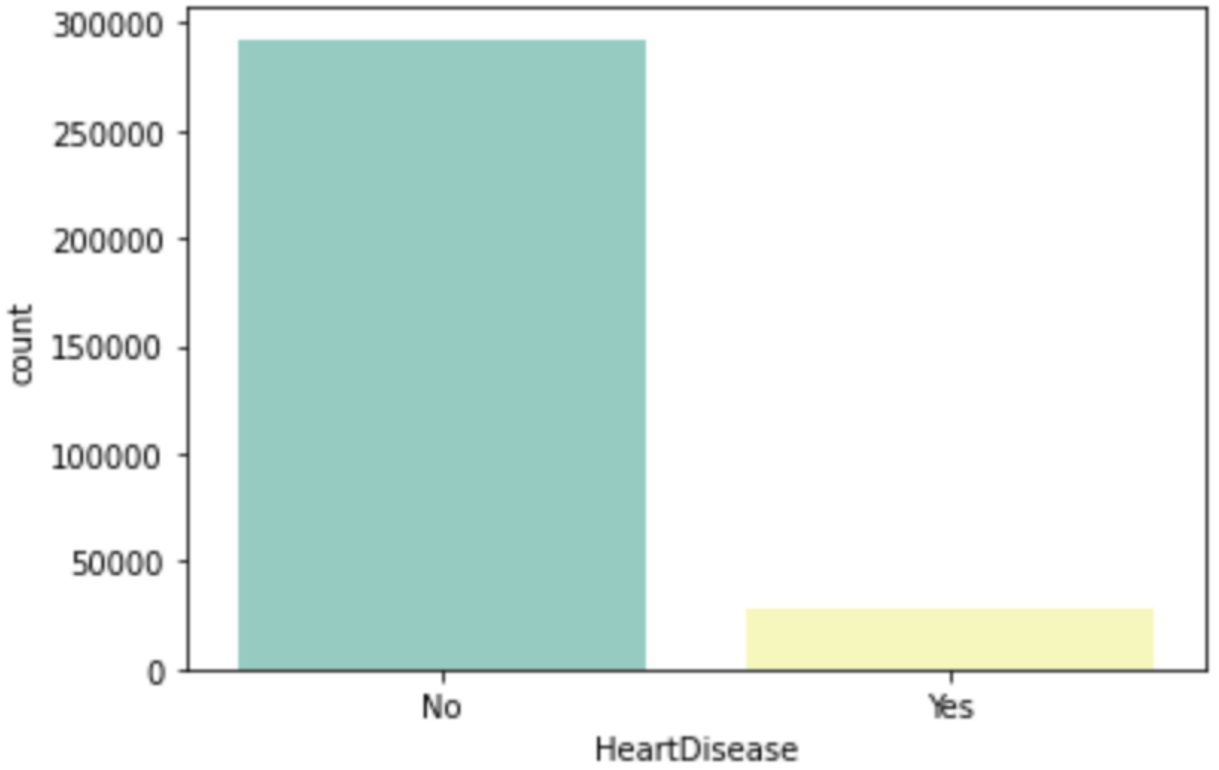
```
msno.bar(train, figsize = (16,5),color = "#7B68EE")
plt.show()
```



Distribution of Target

```
1  sns.countplot(x='HeartDisease',data=train,palette='Set3')
```

```
<AxesSubplot:xlabel='HeartDisease', ylabel='count'>
```



Unique values in categorical objects

```
for column_name in train.columns:
    unique_values = len(train[column_name].unique())
    print("Feature '{column_name}' has '{unique_values}' unique values"\
          .format(column_name = column_name, unique_values=unique_values))
```

```
Feature 'HeartDisease' has '2' unique values
Feature 'BMI' has '3604' unique values
Feature 'Smoking' has '2' unique values
Feature 'AlcoholDrinking' has '2' unique values
Feature 'Stroke' has '2' unique values
Feature 'PhysicalHealth' has '31' unique values
Feature 'MentalHealth' has '31' unique values
Feature 'DiffWalking' has '2' unique values
Feature 'Sex' has '2' unique values
Feature 'AgeCategory' has '13' unique values
Feature 'Race' has '6' unique values
Feature 'Diabetic' has '4' unique values
Feature 'PhysicalActivity' has '2' unique values
Feature 'GenHealth' has '5' unique values
Feature 'SleepTime' has '24' unique values
Feature 'Asthma' has '2' unique values
Feature 'KidneyDisease' has '2' unique values
Feature 'SkinCancer' has '2' unique values
```

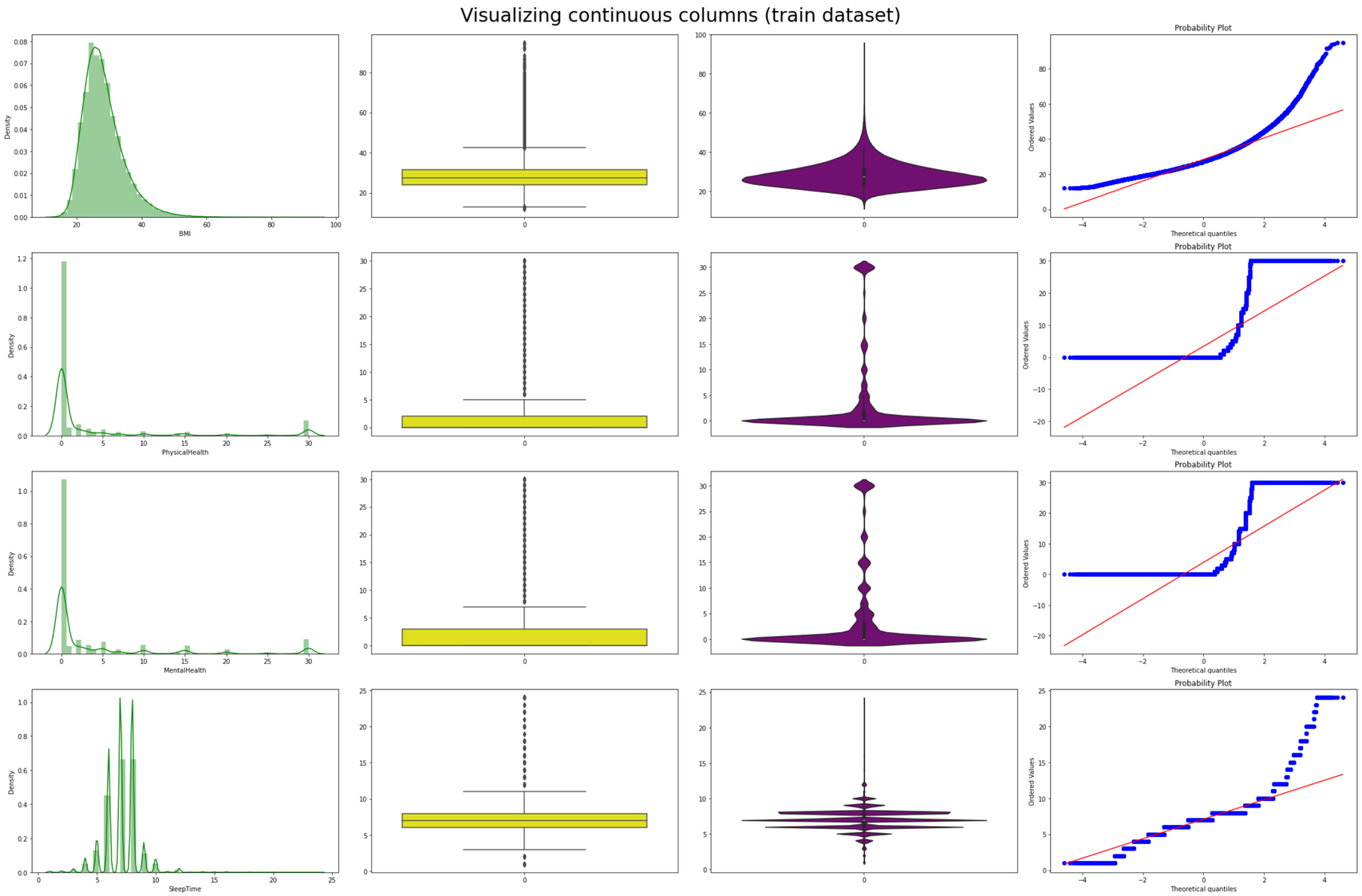
Visualizing continuous columns

```
import warnings
warnings.filterwarnings('ignore')

fig,ax = plt.subplots(len(numeric_features),4,figsize=(30,20))
for index,i in enumerate(numeric_features):
    sns.distplot(train[i],ax=ax[index,0],color='green')
    sns.boxplot(train[i],ax=ax[index,1],color='yellow')
    sns.violinplot(train[i],ax=ax[index,2],color='purple')
    stats.probplot(train[i],plot=ax[index,3])

fig.tight_layout()
fig.subplots_adjust(top=0.95)
plt.suptitle("Visualizing continuous columns (train dataset)",fontsize=30)
```

Visualizing continuous columns

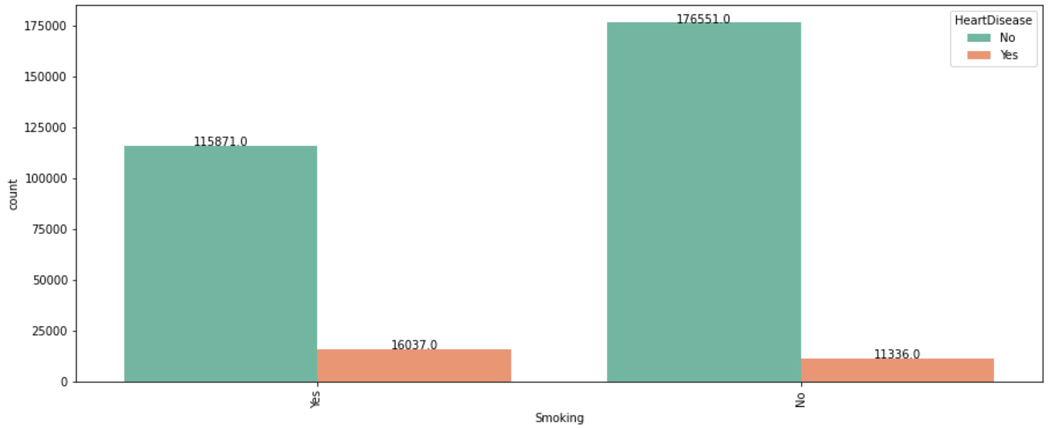


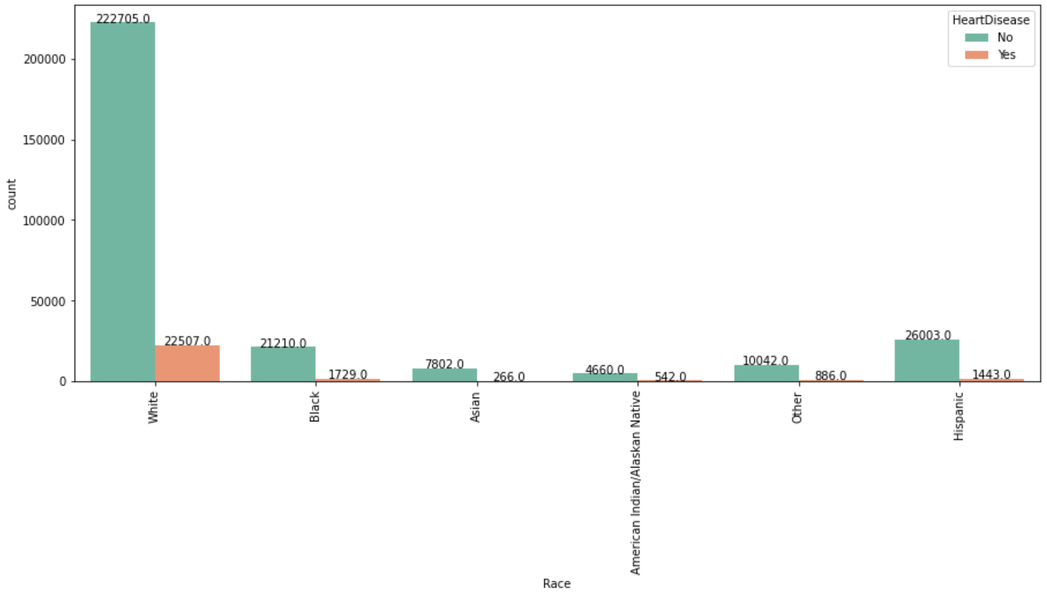
of people with heart disease from various factors

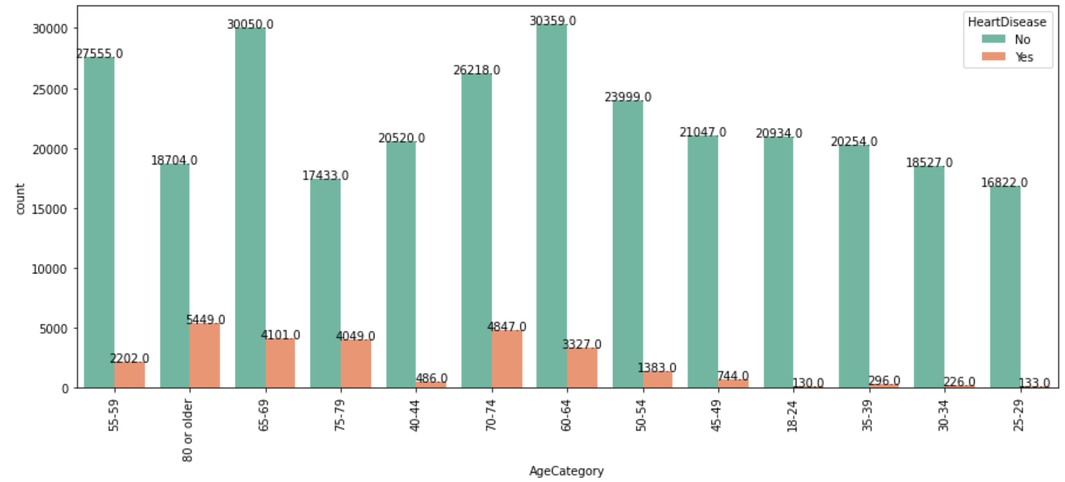
Visualise number of people with heart disease from various factors.

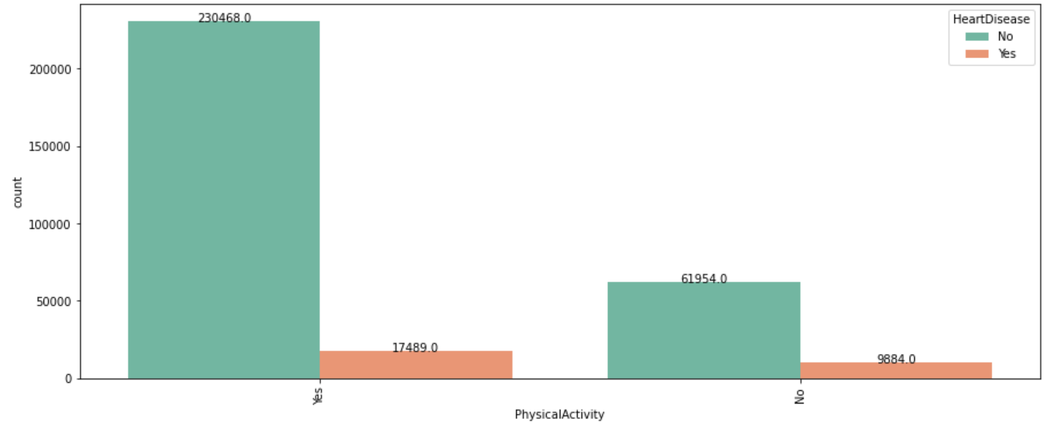
```
for feature in categorical_features:
  fig, ax1 = plt.subplots(figsize=(15,6))
  graph = sns.countplot(ax=ax1,x = feature , data = train,hue='HeartDisease',palette='Set2')
  graph.set_xticklabels(graph.get_xticklabels(),rotation=90)
  for p in graph.patches:
    height = p.get_height()
    graph.text(p.get_x()+p.get_width()/2., height + 0.1,height ,ha="center")
```

Overview part of categorical plots





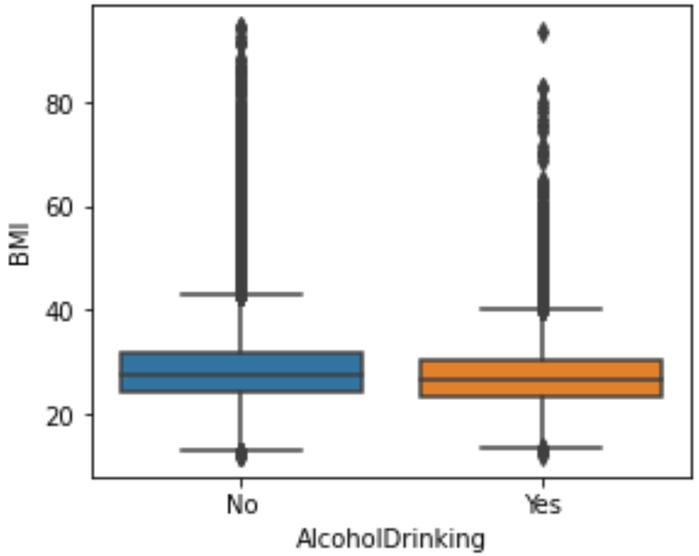


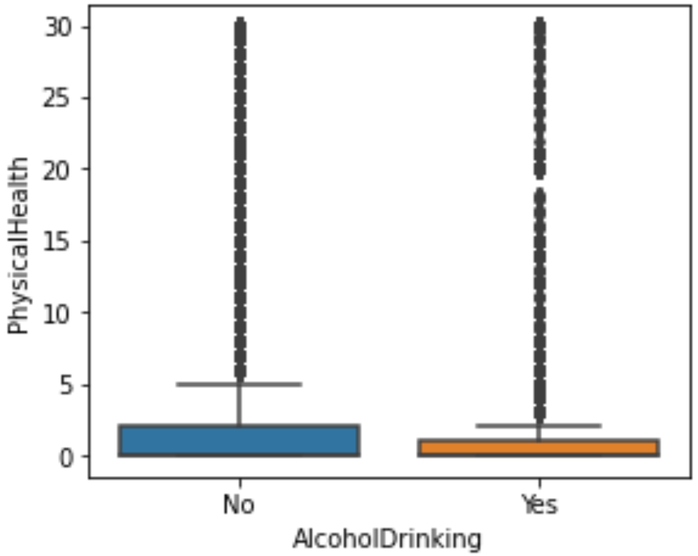


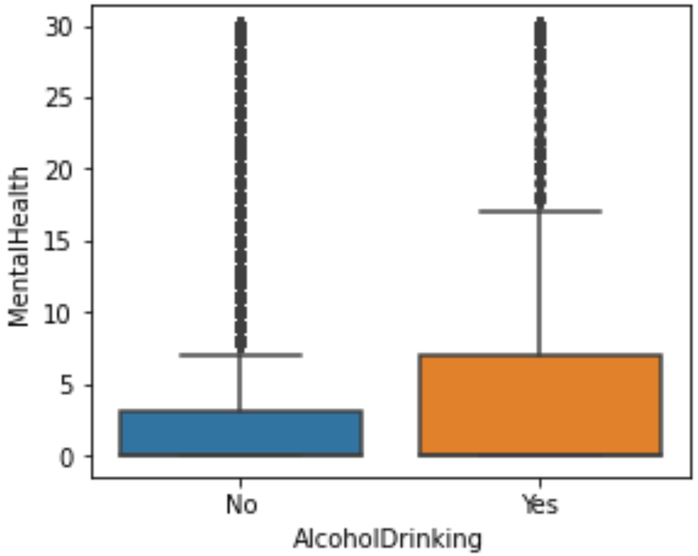
Outliers in numerical variables

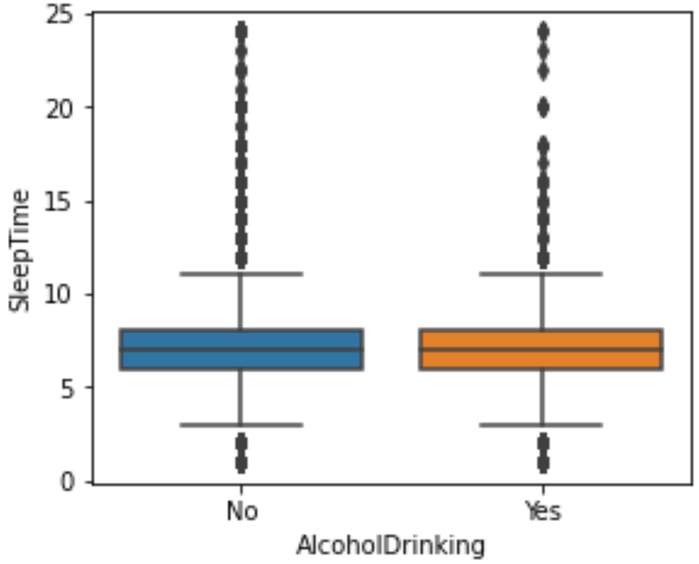
```
plt.figure(figsize=(15,25))
for i,feature in enumerate(numeric_features):
    plt.subplot(6,3,i+1)
    sns.boxplot(y=train[feature], x = train['AlcoholDrinking'])
```

Outliers in numerical variables





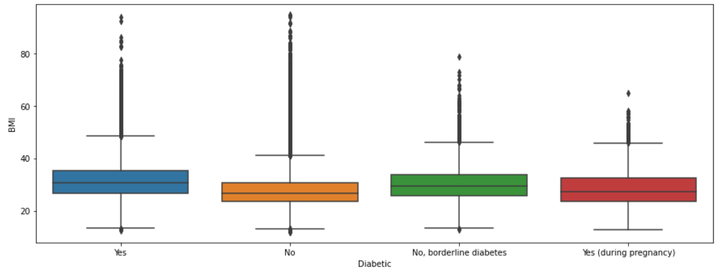


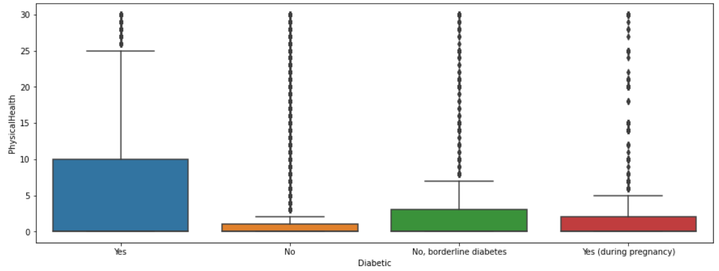


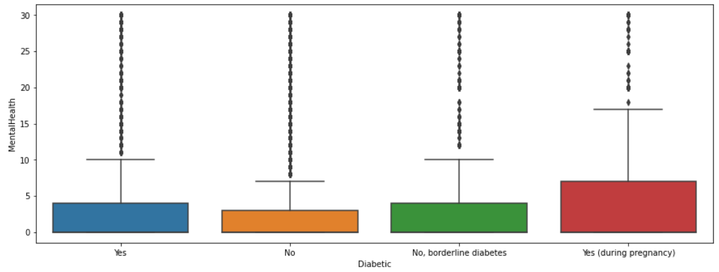
Outliers in numerical variables

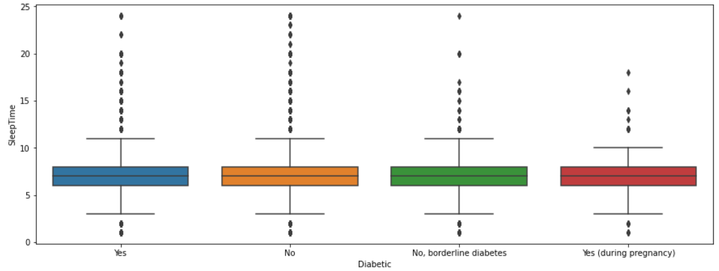
```
plt.figure(figsize=(15,25))
for i,feature in enumerate(numeric_features):
    plt.subplot(4,1,i+1)
    sns.boxplot(y=train[feature], x = train['Diabetic'])
```

Outliers in numerical variables









Test for normality

```python
from scipy.stats import shapiro
# normality test
for feature in numeric_features:
    stat, p = shapiro(train[feature])
    print('Statistics=%.3f, p=%.3f' % (stat, p))
    # interpret
    alpha = 0.05
    if p > alpha:
        print('Sample looks Gaussian (fail to reject H0)')
    else:
        print('Sample does not look Gaussian (reject H0)')
```

```
Statistics=0.928, p=0.000
Sample does not look Gaussian (reject H0)
Statistics=0.476, p=0.000
Sample does not look Gaussian (reject H0)
Statistics=0.551, p=0.000
Sample does not look Gaussian (reject H0)
Statistics=0.892, p=0.000
Sample does not look Gaussian (reject H0)
```

All variables are not normal distributed.

Transform our dataset using the OrdinalEncoder method

Encode categorical features as an integer array.

The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are converted to ordinal integers. This results in a single column of integers (0 to n_categories - 1) per feature.

```
1 from sklearn.preprocessing import OrdinalEncoder
2 enc = OrdinalEncoder()
3 enc.fit(train[categorical_features])
4 train[categorical_features] = enc.transform(train[categorical_features])
```

Build correlation table

```
1 correlation = train.corr()
2 print(correlation['HeartDisease'].sort_values(ascending = False),'\n')
```

```
HeartDisease         1.000000
AgeCategory          0.233432
DiffWalking          0.201258
Stroke               0.196835
PhysicalHealth       0.170721
Diabetic             0.168553
KidneyDisease        0.145197
Smoking              0.107764
SkinCancer           0.093317
Sex                  0.070040
BMI                  0.051803
Asthma               0.041444
Race                 0.034854
MentalHealth         0.028591
SleepTime            0.008327
GenHealth           -0.011062
AlcoholDrinking     -0.032080
PhysicalActivity    -0.100030
Name: HeartDisease, dtype: float64
```

Correlation table

This table is difficult to read and be understood.

```
1 correlation
```

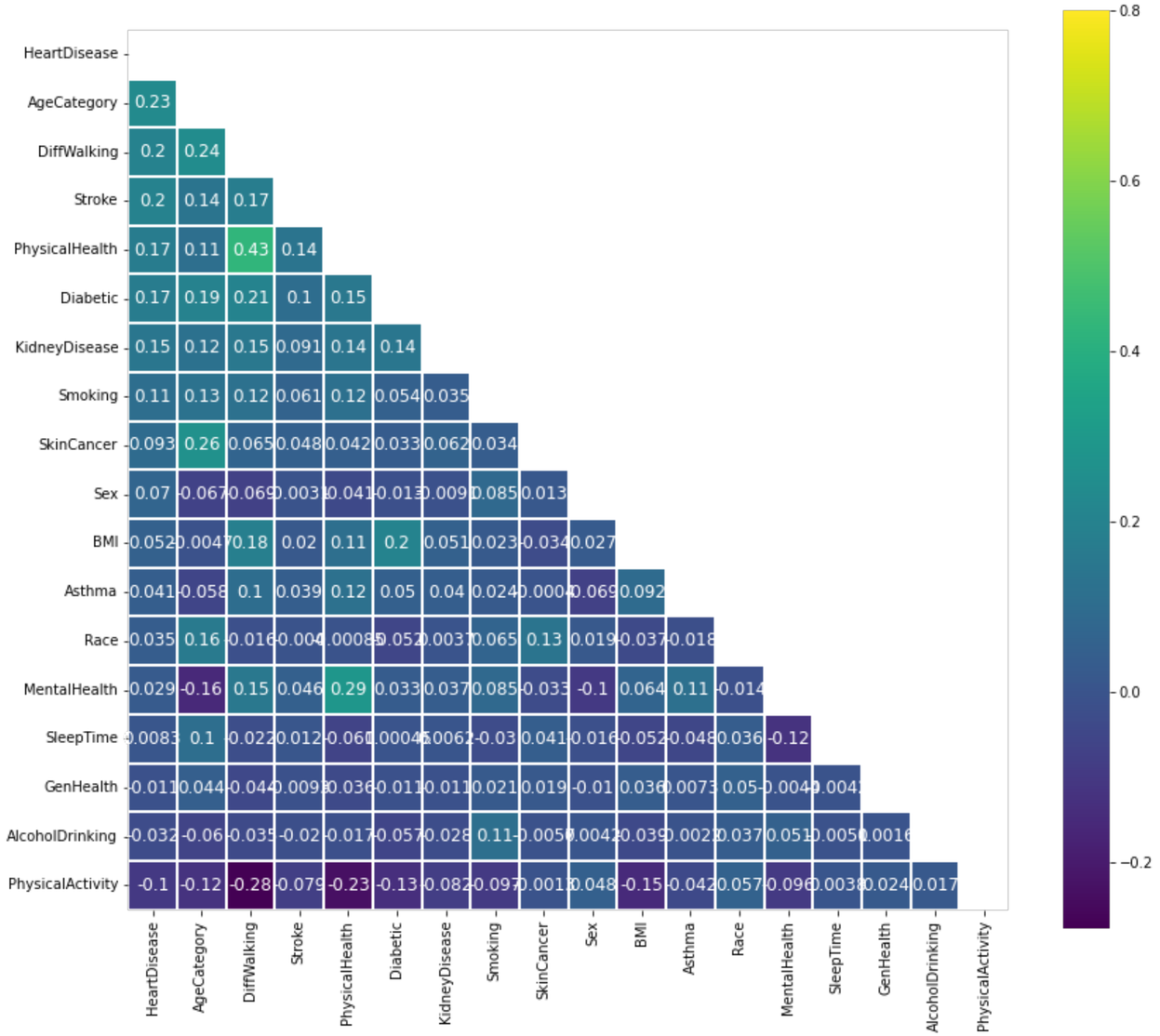
	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalH
HeartDisease	1.000000	0.051803	0.107764	-0.032080	0.196835	0.17
BMI	0.051803	1.000000	0.023118	-0.038816	0.019733	0.1(
Smoking	0.107764	0.023118	1.000000	0.111768	0.061226	0.11
AlcoholDrinking	-0.032080	-0.038816	0.111768	1.000000	-0.019858	-0.01
Stroke	0.196835	0.019733	0.061226	-0.019858	1.000000	0.13
PhysicalHealth	0.170721	0.109788	0.115352	-0.017254	0.137014	1.00
MentalHealth	0.028591	0.064131	0.085157	0.051282	0.046467	0.28

Correlation heatmap

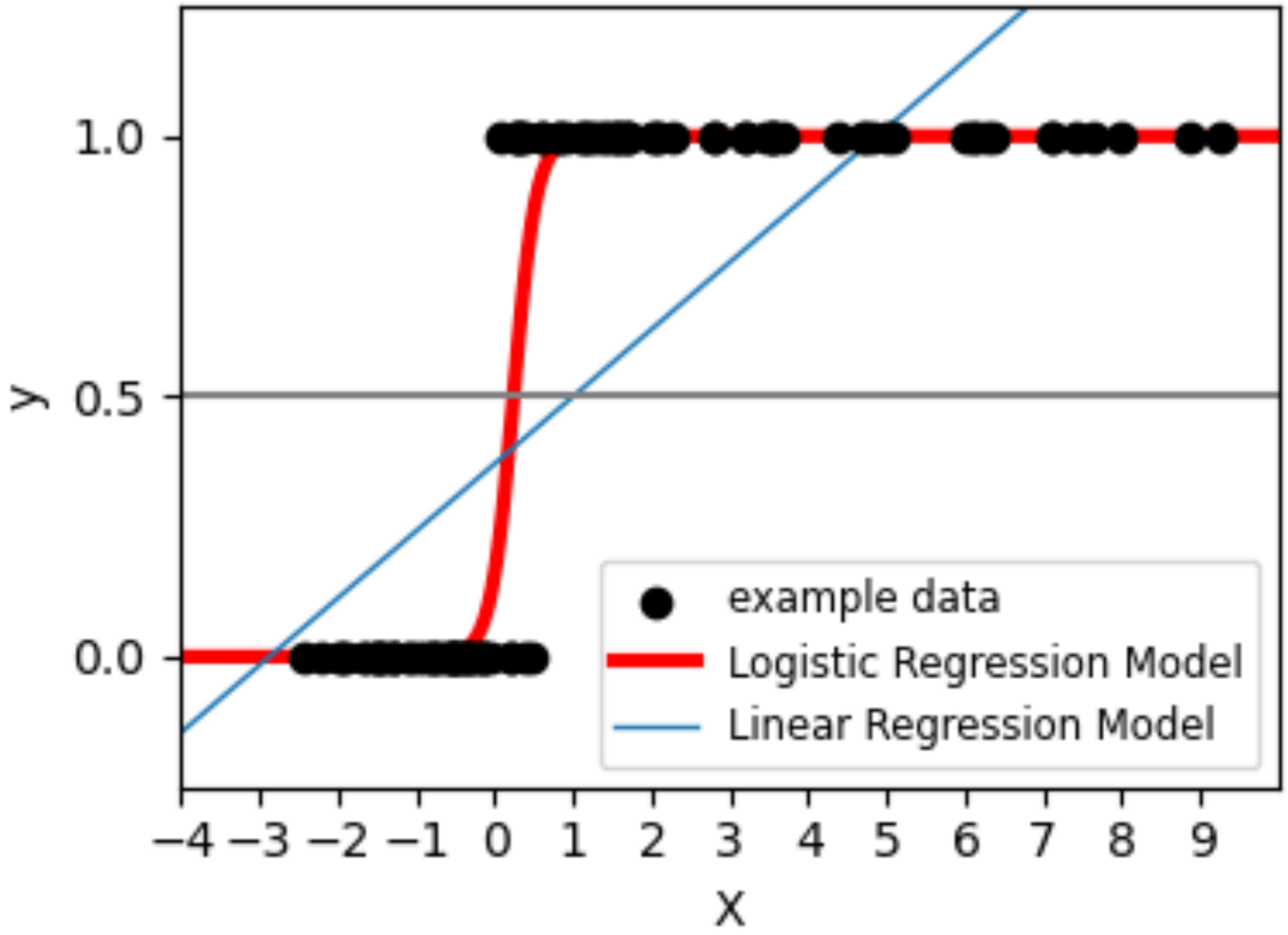
```
k = correlation.shape[1]
cols = correlation.nlargest(k,'HeartDisease')['HeartDisease'].index
print(cols)
cm = np.corrcoef(train[cols].values.T)
mask = np.triu(np.ones_like(train.corr()))
f , ax = plt.subplots(figsize = (16,16))
sns.heatmap(cm,mask=mask, vmax=.8, linewidths=0.01,square=True,annot=True,cmap='viridis',
            linecolor="white",xticklabels = cols.values ,annot_kws = {'size':12},yticklabels = cols.values)
```

```
Index(['HeartDisease', 'AgeCategory', 'DiffWalking', 'Stroke',
       'PhysicalHealth', 'Diabetic', 'KidneyDisease', 'Smoking', 'SkinCancer',
       'Sex', 'BMI', 'Asthma', 'Race', 'MentalHealth', 'SleepTime',
       'GenHealth', 'AlcoholDrinking', 'PhysicalActivity'],
      dtype='object')
```

Correlation heatmap



Understand machine learning



Machine learning

We use KNeighborsClassifier and LogisticRegression method to train model to predict Heart Disease with given features. Then print out the metrics result.

```
from sklearn.model_selection import train_test_split, KFold, GridSearchCV

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score,recall_score
from sklearn.metrics import f1_score
```

```
y=train['HeartDisease']
train.drop('HeartDisease',axis=1,inplace=True)
X_train, X_test, y_train, y_test=train_test_split(train,y,test_size=0.1,random_state=42)
```

Machine learning

```
models = [KNeighborsClassifier(), LogisticRegression()]
scores = dict()

for m in models:
    m.fit(X_train, y_train)
    y_pred = m.predict(X_test)

    print(f'model: {str(m)}')
    print(f'Accuracy_score: {accuracy_score(y_test,y_pred)}')
    print(f'Precission_score: {precision_score(y_test,y_pred)}')
    print(f'Recall_score: {recall_score(y_test,y_pred)}')
    print(f'F1-score: {f1_score(y_test,y_pred)}')
    print('-'*30, '\n')
```

Machine learning

The LogisticRegression trained model had better result in accuracy.

```
model: KNeighborsClassifier()
Accuracy_score: 0.9027517198248906
Precission_score: 0.3271428571428571
Recall_score: 0.079846582984658 3
F1-score: 0.12836322869955158
------------------------------

model: LogisticRegression()
Accuracy_score: 0.9115697310819262
Precission_score: 0.5423728813559322
Recall_score: 0.08926080892608089
F1-score: 0.1532934131736527
------------------------------
```

Retrain LogisticRegression model with all data

We choose to use LogisticRegression model and re-train all the data.

```
1  LR = LogisticRegression()
2  LR.fit(train, y)
```

▼ LogisticRegression

LogisticRegression()

Modelling

Diagnose Heart Disease

```
1 LR.coef_
```

```
array([[ 0.00764391,  0.60000476, -0.52802211,  1.26673338,  0.02173081,
         0.0101688 ,  0.17265137,  0.65981127,  0.27522901, -0.02877603,
         0.30556512, -0.12870659, -0.0520316 , -0.10074024,  0.20310004,
         0.85478629,  0.03180956]])
```

```
1 LR.intercept_
```

```
array([-4.83823486])
```

Prepare a function to output result

```
1 def test_HeartDisease(test):
2     if LR.predict(test)==0:
3         print('You seem healthy and no Heart Disease.')
4     else:
5         print('According to Heart Disease model, \
6 you may be under concerned. Please consult your Dr.')
```

Test with previous data

```
1 test_HeartDisease(train[train.index==5])
```

```
You seem healthy and no Heart Disease.
```

```
1 test_HeartDisease(train[train.index==35])
```

```
According to Heart Disease model, you may be under concerned. Please consult your Dr.
```

```
1 test_HeartDisease(train[train.index==404])
```

```
According to Heart Disease model, you may be under concerned. Please consult your Dr.
```

```
1 test_HeartDisease(train[train.index==3317])
```

```
According to Heart Disease model, you may be under concerned. Please consult your Dr.
```

```
1 test_HeartDisease(train[train.index==1311])
```

```
According to Heart Disease model, you may be under concerned. Please consult your Dr.
```

List out the categorical data changed to numerical

```
for c in categorical_features:
    if c != 'HeartDisease':
        print(c, sorted(origin_data[c].unique()))
        print(c, sorted(train[c].unique()))
```

```
Smoking ['No', 'Yes']
Smoking [0.0, 1.0]
AlcoholDrinking ['No', 'Yes']
AlcoholDrinking [0.0, 1.0]
Stroke ['No', 'Yes']
Stroke [0.0, 1.0]
DiffWalking ['No', 'Yes']
DiffWalking [0.0, 1.0]
Sex ['Female', 'Male']
Sex [0.0, 1.0]
AgeCategory ['18-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54', '55-59', '60-64', '65-69', '70-74', '75-7
9', '80 or older']
AgeCategory [0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0]
Race ['American Indian/Alaskan Native', 'Asian', 'Black', 'Hispanic', 'Other', 'White']
Race [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
Diabetic ['No', 'No, borderline diabetes', 'Yes', 'Yes (during pregnancy)']
Diabetic [0.0, 1.0, 2.0, 3.0]
PhysicalActivity ['No', 'Yes']
PhysicalActivity [0.0, 1.0]
GenHealth ['Excellent', 'Fair', 'Good', 'Poor', 'Very good']
GenHealth [0.0, 1.0, 2.0, 3.0, 4.0]
Asthma ['No', 'Yes']
Asthma [0.0, 1.0]
KidneyDisease ['No', 'Yes']
KidneyDisease [0.0, 1.0]
SkinCancer ['No', 'Yes']
SkinCancer [0.0, 1.0]
```

Test with a live data by our model

```
healthy_test = pd.DataFrame({'BMI': 23,
                             'Smoking': 0, # Non smoker
                             'AlcoholDrinking': 0, # No
                             'Stroke':0 , # No
                             'PhysicalHealth': 0, # 0 day injury
                             'MentalHealth': 0, # 0 day not good
                             'DiffWalking': 0, # No
                             'Sex': 1, # Male
                             'AgeCategory':5, # 45-50
                             'Race': 1, # Asian
                             'Diabetic': 0, # no
                             'PhysicalActivity': 1, # Yes, physical activity or exercise
                             'GenHealth': 0, # Excellent
                             'SleepTime': 8,  # 8 hours
                             'Asthma': 0, # No
                             'KidneyDisease': 0, # No
                             'SkinCancer': 0},index=[1]) # No
```

```
test_HeartDisease(healthy_test)
```

```
You seem healthy and no Heart Disease.
```

Chapter Wrap Up

Machine learning involved complicated calculation sometimes, but as a user may not need to take care of everything. Understanding the logic behind and work flow is ready to perform your own modelling.

There are over hundreds of Machine Learning tools. And each tools could be used to coordinate with other ML tools. Such as use multiple ML to predict and combined with weights.

Popular ML tools: scikit-learn, XGBoost, TensorFlow.

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Seaborn:

https://seaborn.pydata.org/examples/index.html

LogisticRegression:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html