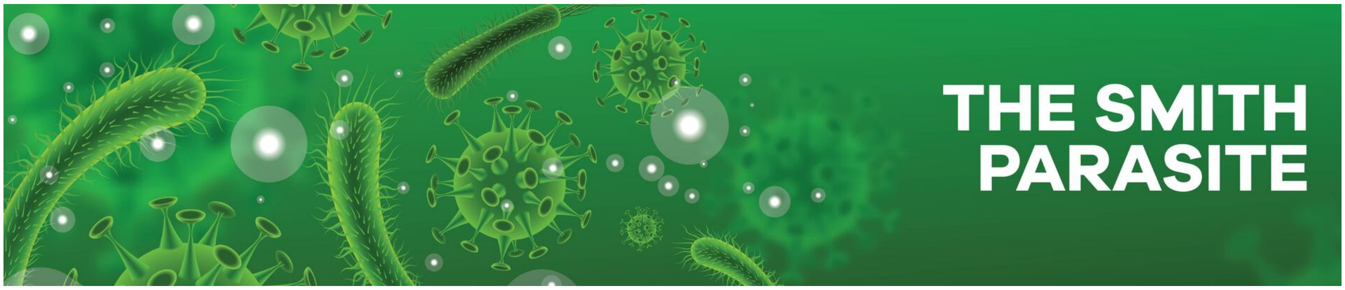


The Smith Parasite - An Unknown Parasitic Disease

Who is more likely to suffer from the Smith Parasite?

First Name	Last Name	ID
Ariel	Cerda	20220662
Gonçalo	Coutinho	20221011
Julio	Vigueras	20220661
Luis	Fernandes	20221649
Miguelangel	Mayuare	20220665

Proposed action plan

For those who have more experience with exploring, cleaning, etc. Please, help with the plan so everybody can follow the same guidelines

This is what I propose:

Steps for the project

1. Frame the problem

This point is pretty clear. Is a classification problem to predict if a list of people have the parasite or not.

2. Explore de data to ge insights

In this step we can use all the techniques required to inform us about the data, which features are useful and which aren't, like visualization, educated assumptions, etc.

3. Prepare the data

All the dropping, dummyfication, transforming and feature engineering.

4. Explore different models and choose the best ones

As it states, here we try the models and use the cross-validation.

5. Fine-tuning and possibly combine the models with some ensemble technique

Using grid search or similar to obtain the best hyper-parameters and if it is better, combine the models.

6. Predict with the competition test set and submit

As the rules says, we can submit 20 times per day, so there is a lot of room for experimentation.

Environment Setup

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# scikit learn imports
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import RandomizedSearchCV

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import make_scorer

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import f1_score

from sklearn.base import BaseEstimator, TransformerMixin

# Visualization packages
import hvplot.pandas
import holoviews as hv
import panel as pn
pd.options.plotting.backend = 'holoviews'

import warnings
warnings.filterwarnings("ignore")

# allows inspection of arguments in a function or method
# delete after finishing the project
import inspect
```

Global settings

In [2]:
```
# seaborn image rendering
```

```python
%config InlineBackend.figure_format = 'retina'

# for showing DataFrames
def tabular(dataframe):
    """From a DataFrame renders an Excel like table

    Args:
        dataframe (DataFrame): DataFrame

    Returns:
        Table: Excel like table
    """
    return dataframe.hvplot.table(selectable=True,
                                  fit_columns=True,
                                  sortable=True,
                                  responsive=True,
                                  width=1080,
                                  index_position=0)


# define a function for countplots

def countplot(df, column1, column2, height=400, stack=True):
    """Groups by two features and makes a countplot with hvplot

    Args:
        df (DataFrame): DataFrame for the plot
        column1 (Pandas Series): First feature to groupby
        column2 (Pandas Series): Second feature to groupby
        height (int, optional): plot's height. Defaults to 400.
        stack (bool, optional): Stack the barplot, Defaults to True.

    Returns:
        holoviews.element.chart.Bars: stacked countplot
    """
    df['Count'] = 1
    grouped = df.groupby([column1, column2]).\
        count()['Count']
    return grouped.hvplot.bar(stacked=stack,
                              rot=45,
                              height=height,
                              title=column1.replace('_', ' '))
```

Read files

In [3]:
```python
# import all the files

train_demo = pd.read_excel('../data/train_demo.xlsx')
train_habits = pd.read_excel('../data/train_habits.xlsx')
train_health = pd.read_excel('../data/train_health.xlsx')
test_demo = pd.read_excel('../data/test_demo.xlsx')
test_habits = pd.read_excel('../data/test_habits.xlsx')
test_health = pd.read_excel('../data/test_health.xlsx')
```

In [4]:
```python
train_demo.head()
```

Out[4]:

	PatientID	Name	Birth_Year	Region	Education	Disease
0	1167	Mrs. Stephanie Gay	1965	London	High School Incomplete (10th to 11th grade)	1
1	1805	Mr. Sherman Nero	1969	South West	High School Incomplete (10th to 11th grade)	1
2	1557	Mr. Mark Boller	1974	Yorkshire and the Humber	Elementary School (1st to 9th grade)	1
3	1658	Mr. David Caffee	1958	London	University Complete (3 or more years)	0
4	1544	Mr. Gerald Emery	1968	South East	University Incomplete (1 to 2 years)	1

In [5]:

```
train_habits.head()
```

Out[5]:

	PatientID	Smoking_Habit	Drinking_Habit	Exercise	Fruit_Habit	Water_Habit
0	1167	No	I usually consume alcohol every day	Yes	Less than 1. I do not consume fruits every day.	Between one liter and two liters
1	1805	No	I consider myself a social drinker	Yes	Less than 1. I do not consume fruits every day.	Between one liter and two liters
2	1557	No	I consider myself a social drinker	No	Less than 1. I do not consume fruits every day.	More than half a liter but less than one liter
3	1658	No	I usually consume alcohol every day	Yes	Less than 1. I do not consume fruits every day.	More than half a liter but less than one liter
4	1544	No	I consider myself a social drinker	No	1 to 2 pieces of fruit in average	More than half a liter but less than one liter

In [6]:

```
train_health.head()
```

Out[6]:

	PatientID	Height	Weight	High_Cholesterol	Blood_Pressure	Mental_Health	Physical_Health	Checkup	Dia
0	1167	155	67	358	120	21	2	More than 3 years	Ne n imm dia
1	1805	173	88	230	142	9	0	Not sure	Ne n imm dia
2	1557	162	68	226	122	26	0	More than 3 years	Ne n imm dia
3	1658	180	66	313	125	13	8	Not sure	I hav preg dia bord
4	1544	180	58	277	125	18	2	More than 3 years	I hav preg dia bord

In [7]:

```
test_demo.head()
```

Out[7]:

	PatientID	Name	Birth_Year	Region	Education
0	1343	Mr. Ricardo Sherman	1970	East Midlands	Elementary School (1st to 9th grade)
1	1727	Mr. Jessie Strickland	1966	Yorkshire and the Humber	University Complete (3 or more years)
2	1828	Mr. Robert Foreman	1978	West Midlands	High School Incomplete (10th to 11th grade)
3	1155	Mr. Edwin Ferguson	1968	Yorkshire and the Humber	High School Incomplete (10th to 11th grade)
4	1020	Mr. Eliseo Krefft	1962	East Midlands	High School Incomplete (10th to 11th grade)

In [8]:

```
test_habits.head()
```

Out[8]:

	PatientID	Smoking_Habit	Drinking_Habit	Exercise	Fruit_Habit	Water_Habit
0	1343	Yes	I usually consume alcohol every day	No	Less than 1. I do not consume fruits every day.	Between one liter and two liters
1	1727	No	I consider myself a social drinker	No	Less than 1. I do not consume fruits every day.	More than half a liter but less than one liter
2	1828	No	I usually consume alcohol every day	Yes	Less than 1. I do not consume fruits every day.	Between one liter and two liters
3	1155	No	I usually consume alcohol every day	No	Less than 1. I do not consume fruits every day.	Less than half a liter
4	1020	No	I consider myself a social drinker	No	Less than 1. I do not consume fruits every day.	Less than half a liter

In [9]:

```
# Join all on ids
data = train_demo.merge(train_habits, how='left')
data = data.merge(train_health, how='left')
data_test = test_demo.merge(test_habits, how='left')
data_test = data_test.merge(test_health, how='left')
data['PatientID'] = data['PatientID'].astype('object')
data_test['PatientID'] = data_test['PatientID'].astype('object')
# data.set_index('PatientID', inplace=True)
# data_test.set_index('PatientID', inplace=True)
```

In [10]:

```
tabular(data)
# data.head()
```

Out[10]:

#	PatientID	Name	Birth_Year	Region	Education	Disease	Smoking_I	Drinking_H	Exercise	Fruit_Habi	Water_Ha	Height
0	1167	Mrs. Steph	1,965	London	High Scho	1	No	I usually co	Yes	Less than	Between o	155
1	1805	Mr. Sherm	1,969	South Wes	High Scho	1	No	I consider	Yes	Less than	Between o	173
2	1557	Mr. Mark E	1,974	Yorkshire a	Elementar	1	No	I consider	No	Less than	More than	162
3	1658	Mr. David (	1,958	London	University	0	No	I usually co	Yes	Less than	More than	180
4	1544	Mr. Gerald	1,968	South Eas	University	1	No	I consider	No	1 to 2 piec	More than	180
5	1653	Mr. David I	1,966	East Midla	nan	0	Yes	I consider	Yes	Less than	More than	167
6	1422	Mrs. Patric	1,965	Yorkshire a	High Scho	1	No	I usually co	Yes	Less than	Less than	158
7	1806	Mr. Wesley	1,965	West Midla	High Scho	0	No	I consider	Yes	1 to 2 piec	More than	178
8	1703	Mr. Billy Ki	1,965	East of En	High Scho	1	No	I usually co	Yes	Less than	Between o	162
9	1370	Mrs. Tina I	1,979	East Midla	High Scho	0	Yes	I consider	Yes	Less than	Between o	154
10	2019	Mr. William	1,975	South Wes	University	1	No	I usually co	No	Less than	Between o	167

In [11]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 0 to 799
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   PatientID         800 non-null    object
 1   Name              800 non-null    object
 2   Birth_Year        800 non-null    int64
 3   Region            800 non-null    object
 4   Education         787 non-null    object
 5   Disease           800 non-null    int64
 6   Smoking_Habit     800 non-null    object
 7   Drinking_Habit    800 non-null    object
 8   Exercise          800 non-null    object
 9   Fruit_Habit       800 non-null    object
 10  Water_Habit       800 non-null    object
 11  Height            800 non-null    int64
 12  Weight            800 non-null    int64
 13  High_Cholesterol  800 non-null    int64
 14  Blood_Pressure    800 non-null    int64
 15  Mental_Health     800 non-null    int64
 16  Physical_Health   800 non-null    int64
 17  Checkup           800 non-null    object
 18  Diabetes          800 non-null    object
dtypes: int64(8), object(11)
memory usage: 125.0+ KB
```

In [12]:
```
# data_test.head()
tabular(data_test)
```

Out[12]:

#	PatientID	Name	Birth_Year	Region	Education	Smoking_H	Drinking_H	Exercise	Fruit_Habit	Water_Hab	Height
0	1343	Mr. Ricardo	1,970	East Midlar	Elementary	Yes	I usually co	No	Less than 1	Between or	172
1	1727	Mr. Jessie :	1,966	Yorkshire a	University (	No	I consider r	No	Less than 1	More than l	171
2	1828	Mr. Robert	1,978	West Midla	High Schoo	No	I usually co	Yes	Less than 1	Between or	171
3	1155	Mr. Edwin I	1,968	Yorkshire a	High Schoo	No	I usually co	No	Less than 1	Less than l	174
4	1020	Mr. Eliseo I	1,962	East Midlar	High Schoo	No	I consider r	No	Less than 1	Less than l	172
5	1751	Mrs. Cristir	1,957	East Midlar	High Schoo	No	I consider r	No	Less than 1	More than l	167
6	1814	Mr. Victor F	1,960	North East	Elementary	Yes	I consider r	No	Less than 1	Between or	174
7	1460	Mr. Robert	1,953	East Midlar	Elementary	No	I consider r	No	5 to 6 piece	Between or	173
8	1913	Mr. Clinton	1,977	South Wes	High Schoo	No	I consider r	Yes	Less than 1	Between or	173
9	1530	Mrs. Cora I	1,962	South Wes	University (	No	I consider r	No	3 to 4 piece	More than l	166
10	1455	Mr. Chase	1,953	South Wes	University (	Yes	I consider r	No	1 to 2 piece	More than l	174

In [13]:
```
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 225 entries, 0 to 224
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   PatientID         225 non-null    object
 1   Name              225 non-null    object
 2   Birth_Year        225 non-null    int64 
 3   Region            225 non-null    object
 4   Education         225 non-null    object
 5   Smoking_Habit     225 non-null    object
 6   Drinking_Habit    225 non-null    object
 7   Exercise          225 non-null    object
 8   Fruit_Habit       225 non-null    object
 9   Water_Habit       225 non-null    object
 10  Height            225 non-null    int64 
 11  Weight            225 non-null    int64 
 12  High_Cholesterol  225 non-null    int64 
 13  Blood_Pressure    225 non-null    int64 
 14  Mental_Health     225 non-null    int64 
 15  Physical_Health   225 non-null    int64 
 16  Checkup           225 non-null    object
 17  Diabetes          225 non-null    object
dtypes: int64(7), object(11)
memory usage: 33.4+ KB
```

Exploratory Data Analysis

Data description

The target variable only have two outcomes, 0 or 1, representing infected or not infected.

All the data exploration, cleaning, preparation and modeling will be done for classification.

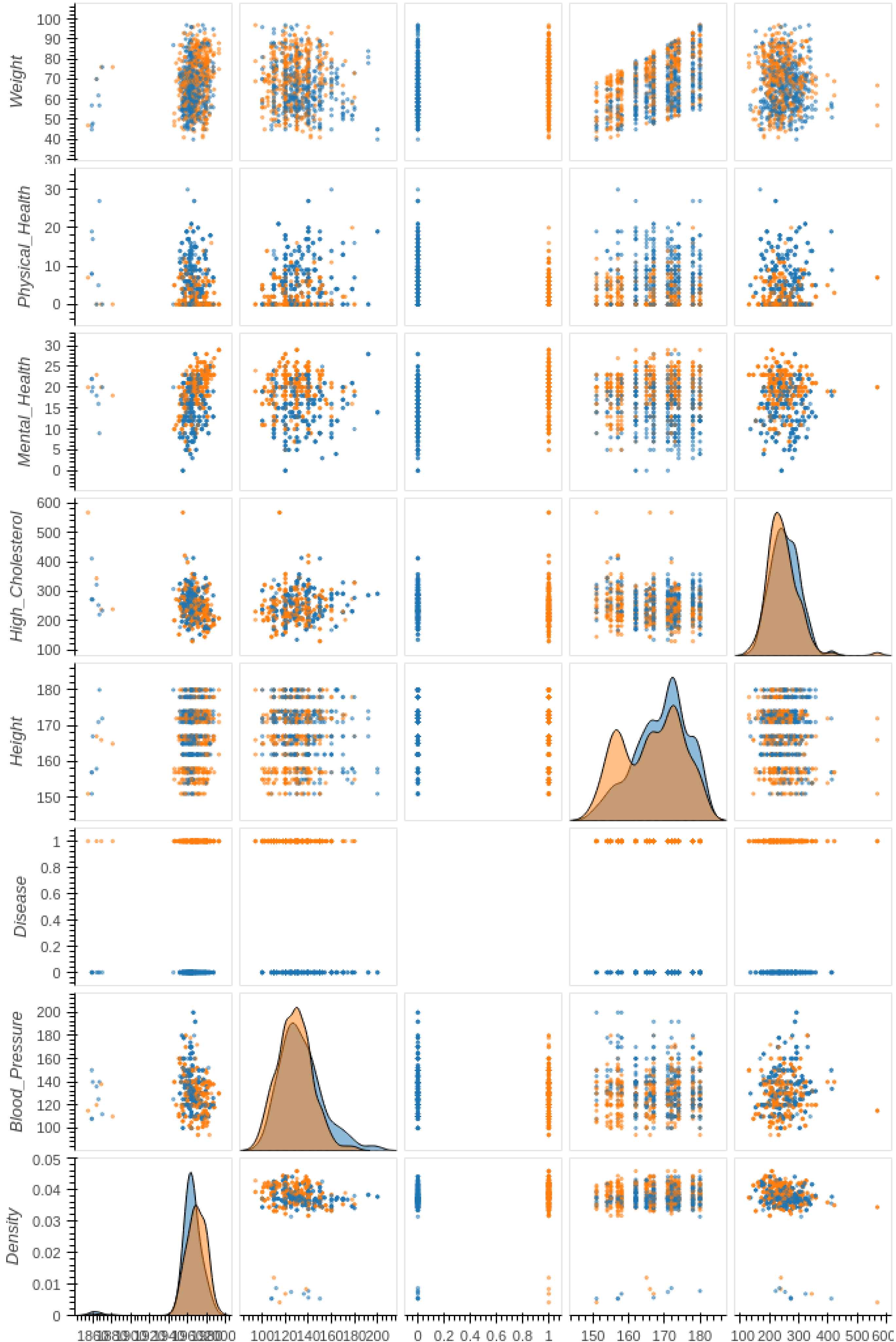
In the scatter matrix below, it can be seen that some distributions differ between infected and not infected, mainly, `Mental_Health`, `Physical_health`, and `Weight`.

In [14]:
```
# first check the balance of the target variable
data['Disease'].value_counts(normalize=True)
```

Out[14]:
```
1    0.51375
0    0.48625
Name: Disease, dtype: float64
```

In [15]:
```
# the difference in colors are infected vs not infected
hvplot.plotting.scatter_matrix(data, c='Disease', alpha=0.5,
                               diagonal='kde')
```

Out[15]:



Weight
Physical_Health
Mental_Health
High_Cholesterol
Height
Disease
Blood_Pressure
Density

Birth_Year Blood_Pressure Disease Height High_Cholesterol

Data Types

All data types seems coherent

In [16]:
```
data.info()
```

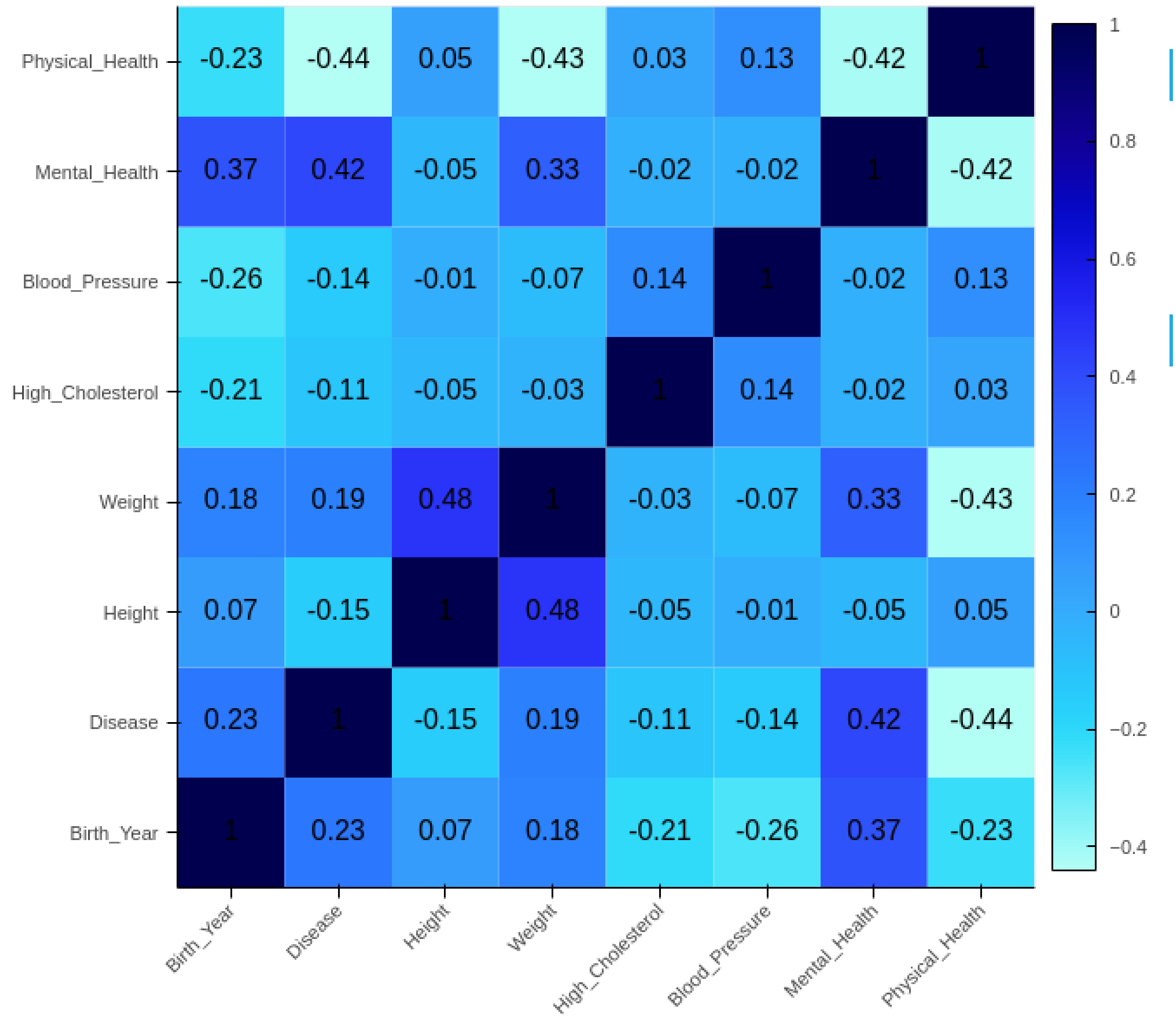
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 0 to 799
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   PatientID         800 non-null    object
 1   Name              800 non-null    object
 2   Birth_Year        800 non-null    int64 
 3   Region            800 non-null    object
 4   Education         787 non-null    object
 5   Disease           800 non-null    int64 
 6   Smoking_Habit     800 non-null    object
 7   Drinking_Habit    800 non-null    object
 8   Exercise          800 non-null    object
 9   Fruit_Habit       800 non-null    object
 10  Water_Habit       800 non-null    object
 11  Height            800 non-null    int64 
 12  Weight            800 non-null    int64 
 13  High_Cholesterol  800 non-null    int64 
 14  Blood_Pressure    800 non-null    int64 
 15  Mental_Health     800 non-null    int64 
 16  Physical_Health   800 non-null    int64 
 17  Checkup           800 non-null    object
 18  Diabetes          800 non-null    object
dtypes: int64(8), object(11)
memory usage: 125.0+ KB
```

Below, we can see the correlation matrix of numerical values using *spearman* correlation.

With this visualization, it can be seen that the aforementioned features seems to be the ones with higher importances.

In [17]:
```
corr_matrix = data.select_dtypes('number').corr(method='spearman')\
    .apply(lambda x: round(x, 2))
heatmap = corr_matrix.hvplot.heatmap(rot=45, height=600)
heatmap * hv.Labels(heatmap)
```

Out[17]:



Categorical features

The feature `Name` can be safely removed. Some features have binary values, like `Smoking_Habits` and `Exercise` while others require further exploration.

In [18]:

```
cat = data.select_dtypes('object')
cat['Disease'] = data['Disease']
cat.head()
```

Out[18]:

	PatientID	Name	Region	Education	Smoking_Habit	Drinking_Habit	Exercise	Fruit_Habit	Water_Habi
0	1167	Mrs. Stephanie Gay	London	High School Incomplete (10th to 11th grade)	No	I usually consume alcohol every day	Yes	Less than 1. I do not consume fruits every day.	Betweer one liter anc two liters
1	1805	Mr. Sherman Nero	South West	High School Incomplete (10th to 11th grade)	No	I consider myself a social drinker	Yes	Less than 1. I do not consume fruits every day.	Betweer one liter anc two liters
2	1557	Mr. Mark Boller	Yorkshire and the Humber	Elementary School (1st to 9th grade)	No	I consider myself a social drinker	No	Less than 1. I do not consume fruits every day.	More thar half a lite but less thar one lite
3	1658	Mr. David Caffee	London	University Complete (3 or more years)	No	I usually consume alcohol every day	Yes	Less than 1. I do not consume fruits every day.	More thar half a lite but less thar one lite
4	1544	Mr. Gerald Emery	South East	University Incomplete (1 to 2 years)	No	I consider myself a social drinker	No	1 to 2 pieces of fruit in average	More thar half a lite but less thar one lite

For regions, two values for London can be seen in the barplot below, the first instance is to fix it before exploration.

In [19]:
```python
cat['Region'].value_counts()
```

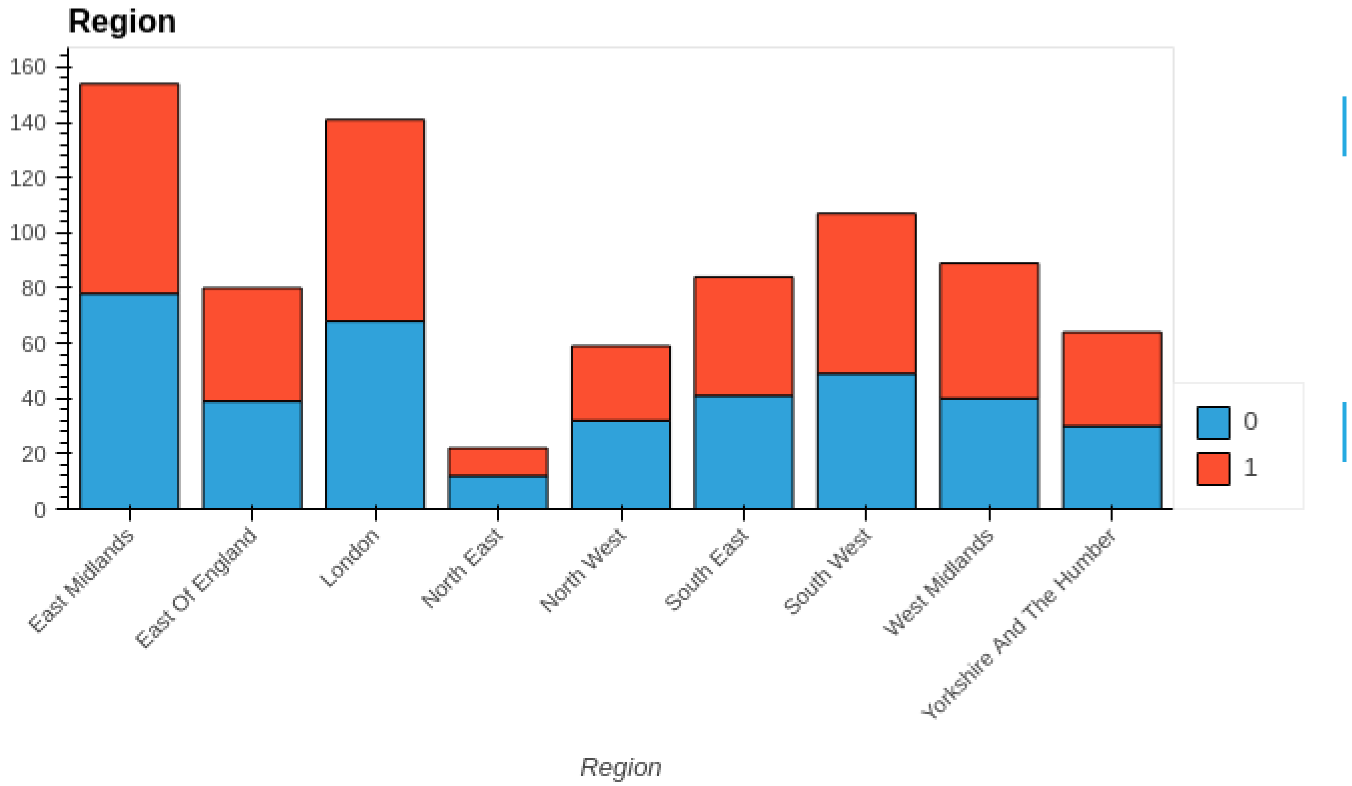
Out[19]:
```
East Midlands               154
London                      136
South West                  107
West Midlands                89
South East                   84
East of England              80
Yorkshire and the Humber     64
North West                   59
North East                   22
LONDON                        5
Name: Region, dtype: int64
```

Visual inspection

Below are the countplots of every categorical feature. Blue means not infected and red infected.

In [20]:
```python
cat['Region'] = data['Region'].str.title()
cat['Count'] = 1
countplot(cat, 'Region', 'Disease')
```

Out[20]:



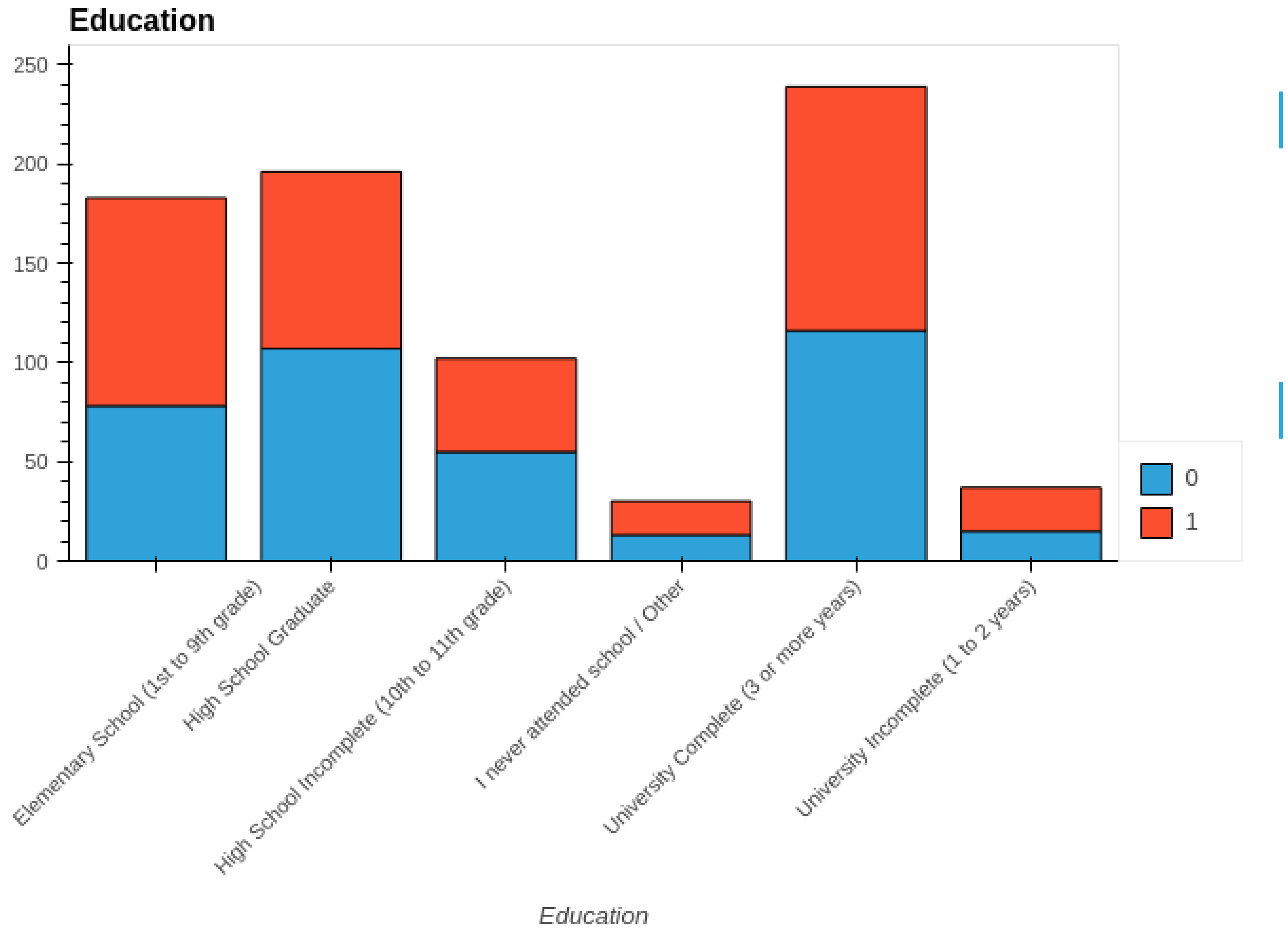
Region

Almost in every region, there is a proportion of 1:1 of the target. There seems that the parasite infections are widespread and do not clustered around certain regions.

Considering the low effect, this feature could be discarded.

In [21]:
```
countplot(cat, 'Education', 'Disease', 500)
```

Out[21]:

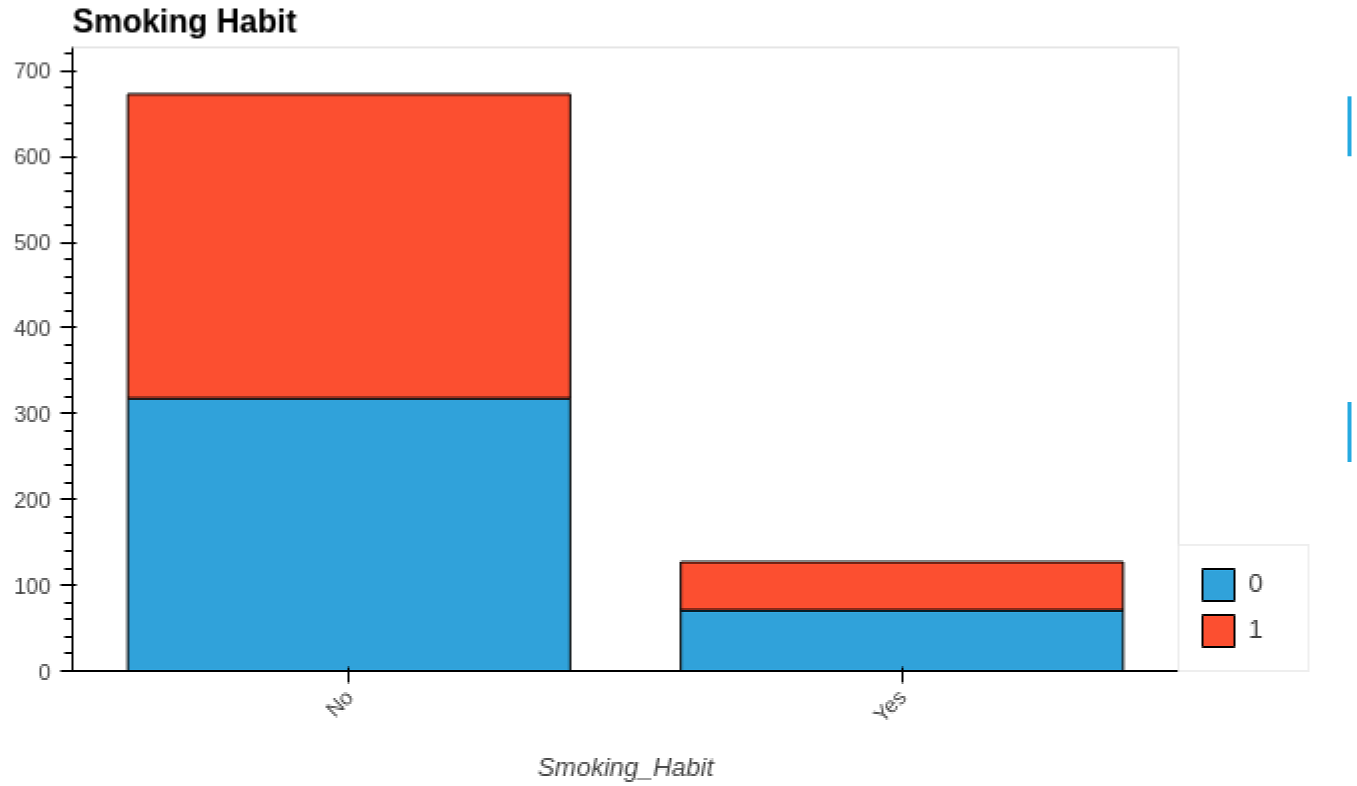


Education

Having certain level of education, has little effect on the infections, only people with very low education, Elementary School, seems to be affected.

In [22]:
```
countplot(cat, 'Smoking_Habit', 'Disease')
```

Out[22]:



Smoking Habits

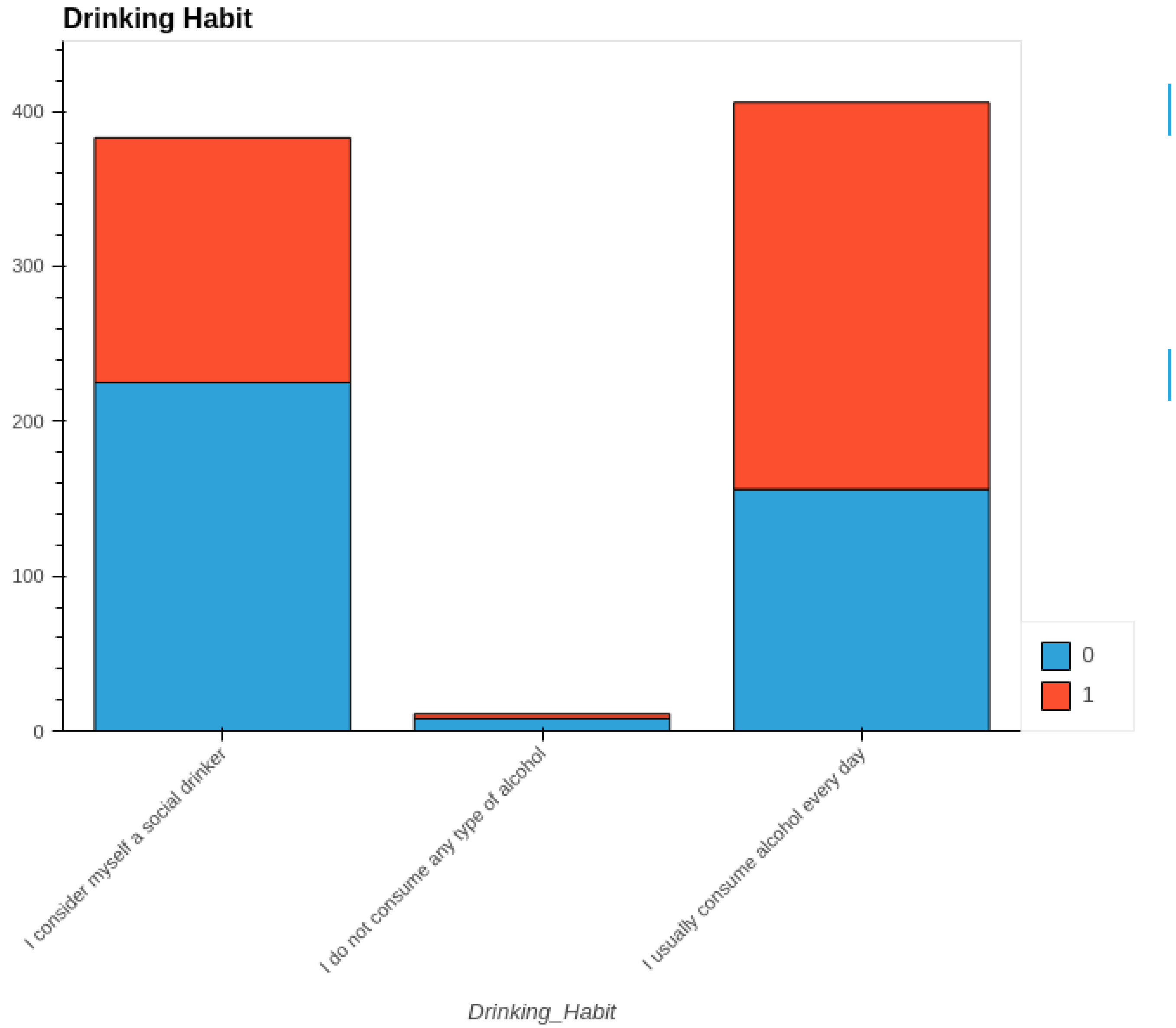
Non smokers show no advantage over smokers in the plot. Both groups are equally likely to get infected.

This feature could be discarded.

In [23]:

```
countplot(cat, 'Drinking_Habit', 'Disease', 600)
```

Out[23]:



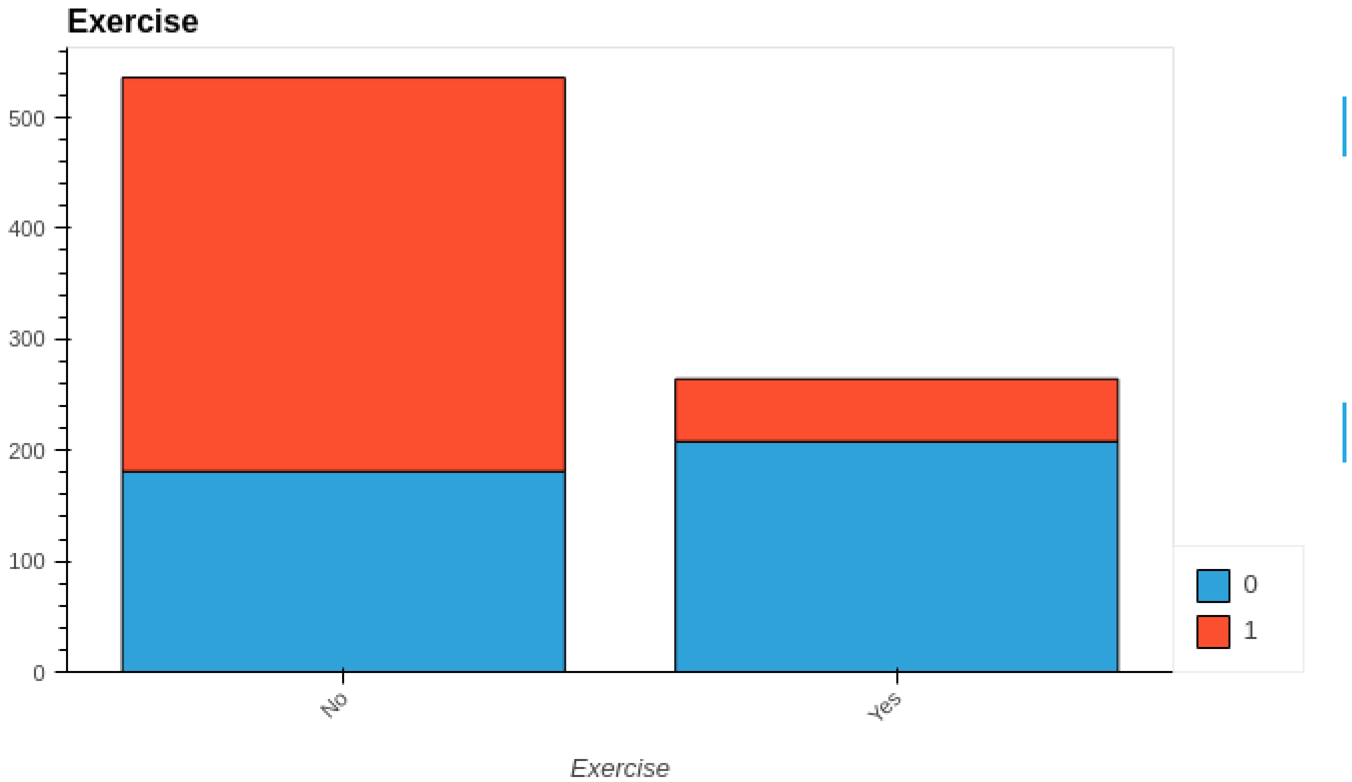
Drinking Habits

There is a difference with this habits. Frequent drinkers tend to be more prone to be infected than social drinkers. "Not consumers" shows the opposite trend, nevertheless, the sample is very small and can be due prure randomness.

In [24]:

```
countplot(cat, 'Exercise', 'Disease')
```

Out[24]:



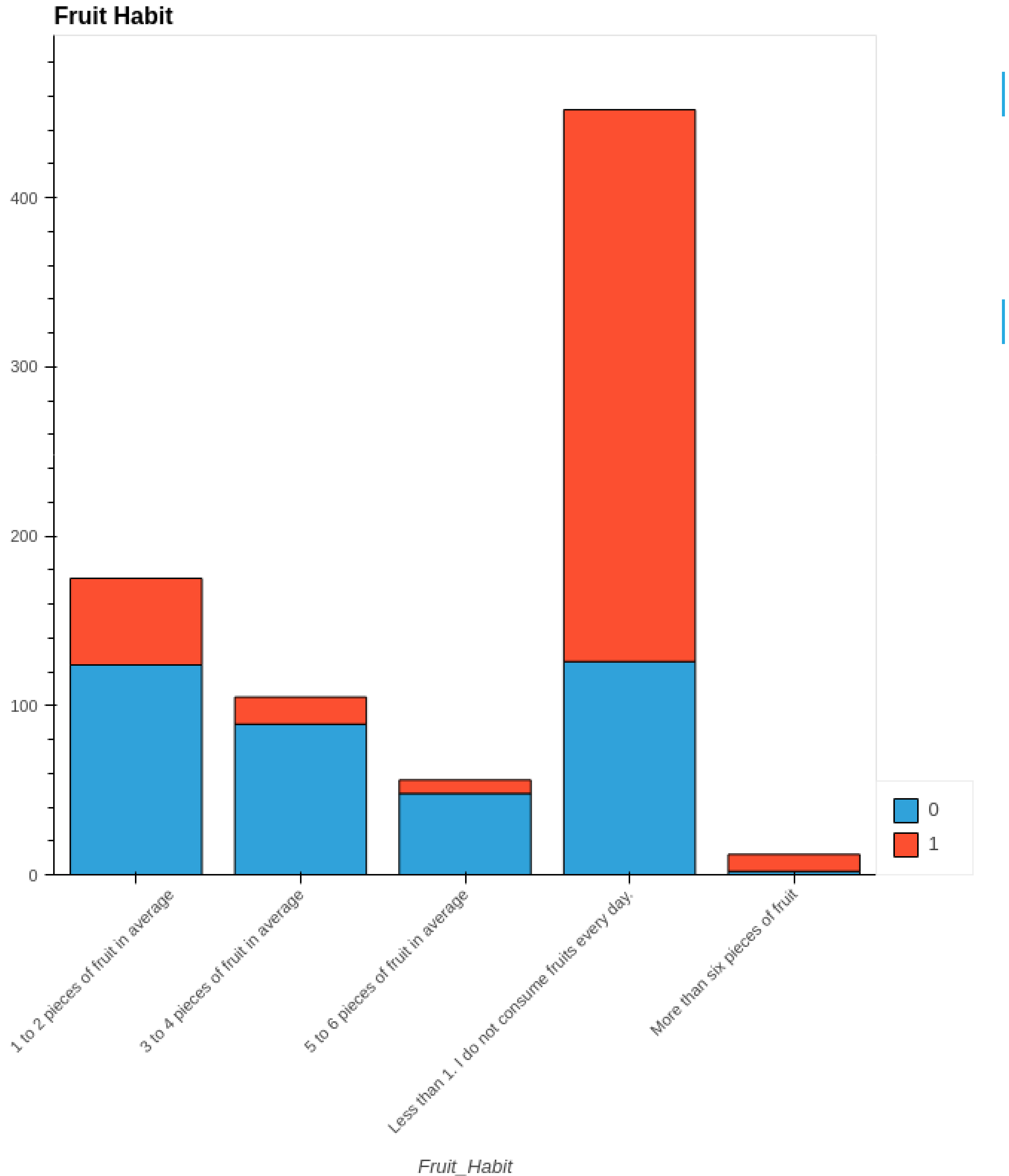
Exercise

People who workout more than 3 weeks per day are less prone to be infected, the other group shows the opposite trend.

In [25]:

```
countplot(cat, 'Fruit_Habit', 'Disease', 800)
```

Out[25]:



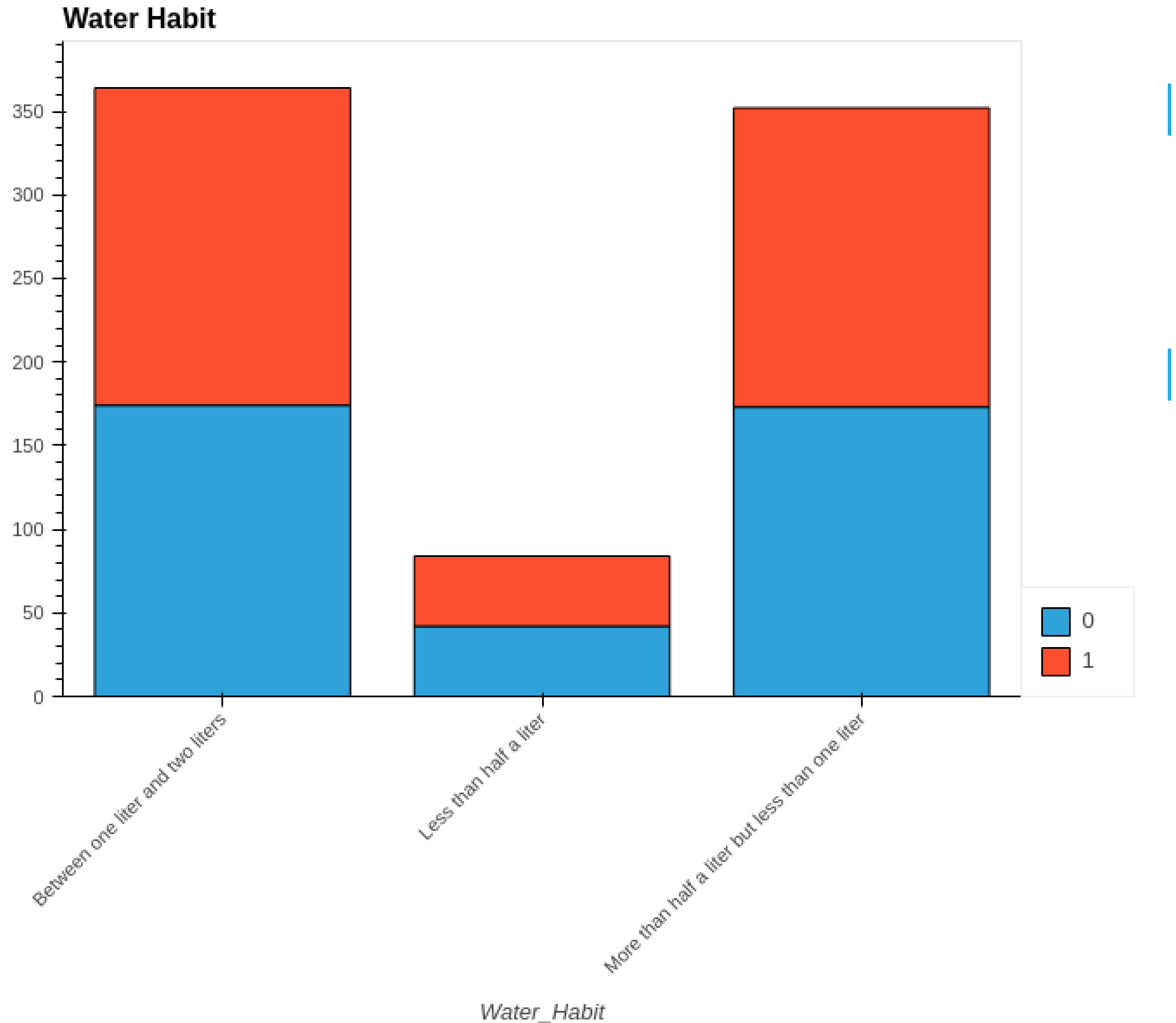
Fruit Habits

The trend is clear, people that eats fruits more regularly is less likely to be infected. It seems eating habits play a major role. The exception is the group that eat more than 6 fruits a day that can be product of randomness considering the group size.

This feature could be encoded in an ordinal manner.

In [26]:
```
countplot(cat, 'Water_Habit', 'Disease', 600)
```

Out[26]:



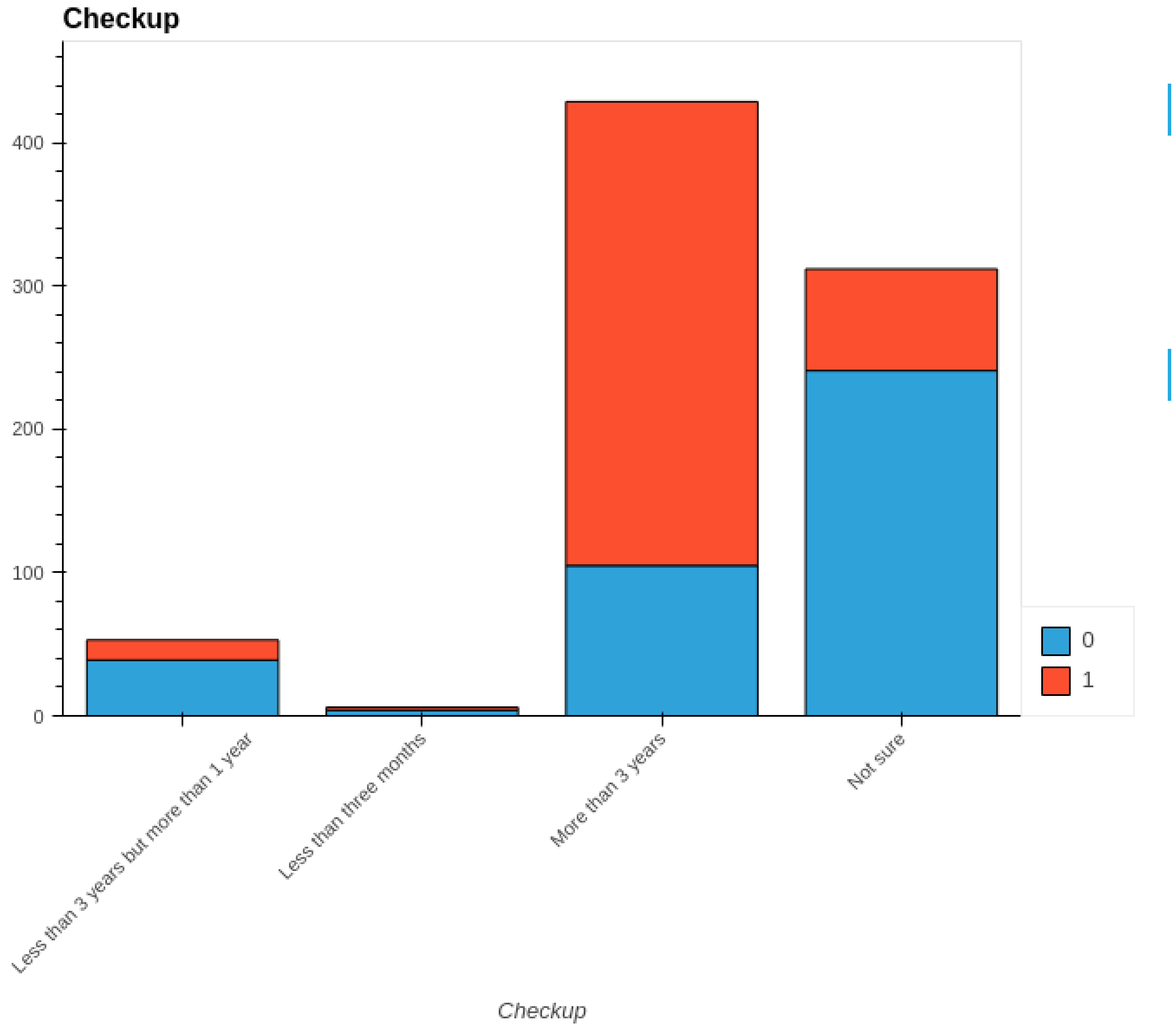
Water Habits

Water habits shows no importance to determine the infection outcome in the individuals.

This feature could be discarded.

In [27]:
```
countplot(cat, 'Checkup', 'Disease', 600)
```

Out[27]:

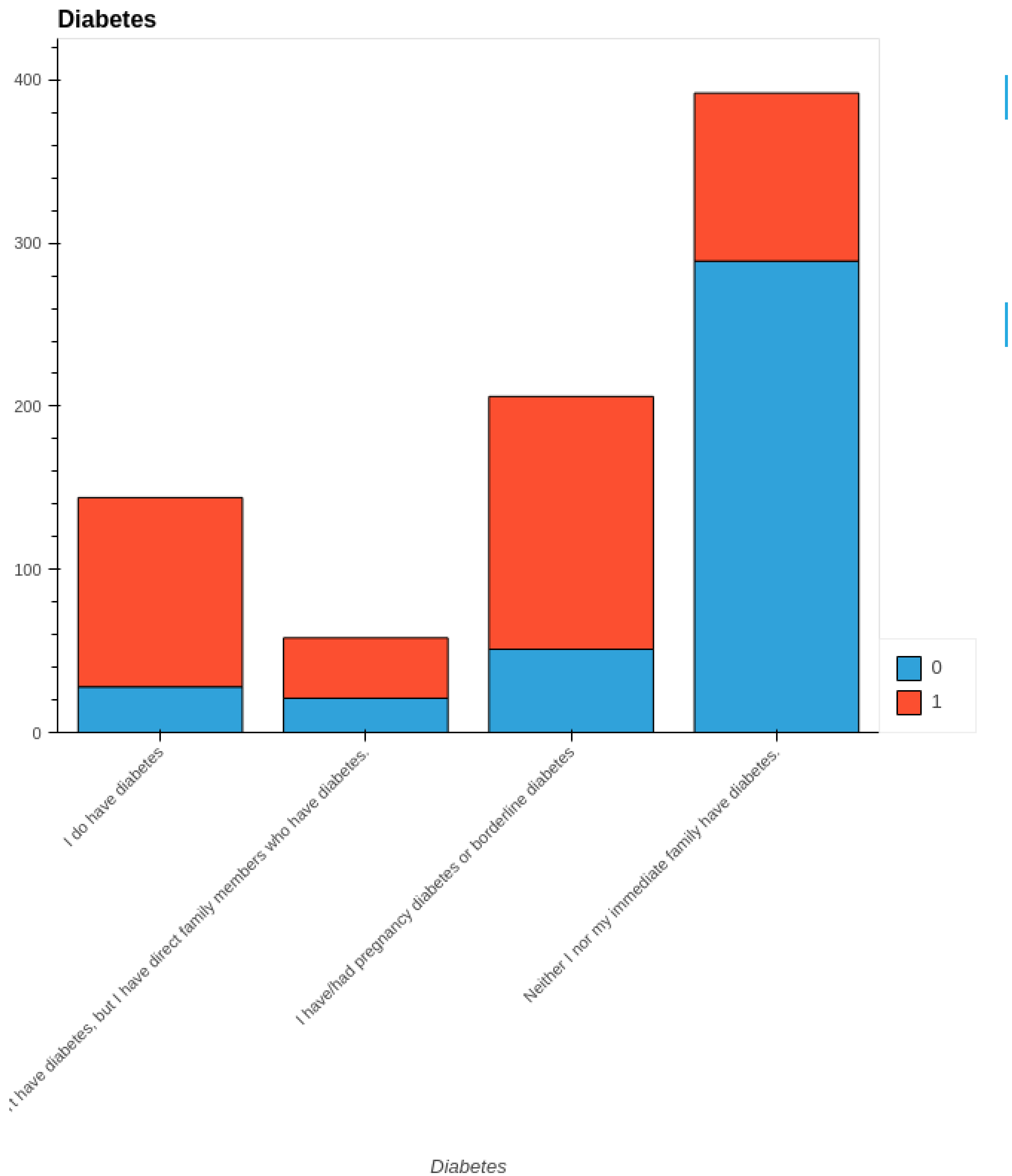


Medical Checkup

People whom are certain of no having checkups in more than 3 years are more prone to be infected. Health neglency can be the cause but that can derive in other causes like. food habits (Fruits in our data).

In [28]:
```
countplot(cat, 'Diabetes', 'Disease', 800)
```

Out[28]:



Diabetes

The trend is clear, diabetics are more likely to get infected and is a clear risk factor.

People with direct family with diabetes seems to be also a risk factor. This group tend to have the same living habits as their direct family wich can lead to similar outcomes.

Possible features to drop: `['Name', 'Region', 'Education', 'Smoking_Habit', 'Water_Habit']`.

In [29]:

```
to_drop = ['Name', 'Region', 'Education', 'Smoking_Habit', 'Water_Habit']
```

Numerical Data Distribution

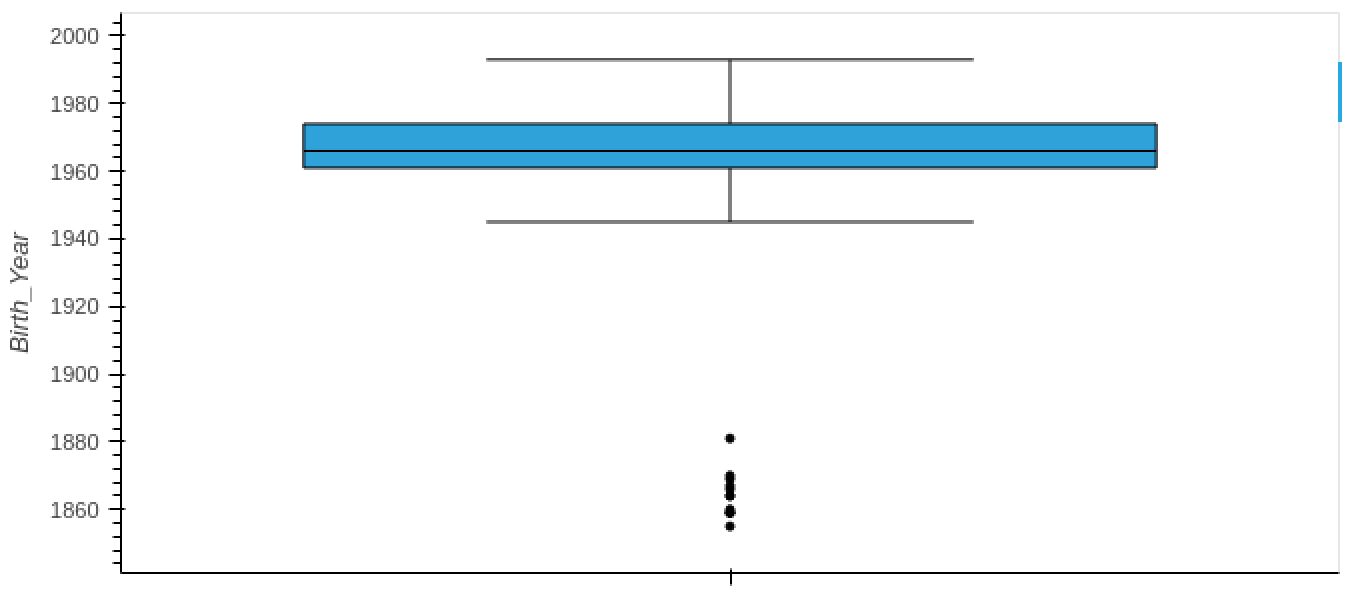
In [30]:
```
data.describe()
```

Out[30]:

	Birth_Year	Disease	Height	Weight	High_Cholesterol	Blood_Pressure	Mental_Health	Phys
count	800.000000	800.000000	800.000000	800.00000	800.000000	800.000000	800.000000	
mean	1966.043750	0.513750	167.806250	67.82750	249.322500	131.053750	17.345000	
std	15.421872	0.500124	7.976888	12.11347	51.566631	17.052693	5.385139	
min	1855.000000	0.000000	151.000000	40.00000	130.000000	94.000000	0.000000	
25%	1961.000000	0.000000	162.000000	58.00000	213.750000	120.000000	13.000000	
50%	1966.000000	1.000000	167.000000	68.00000	244.000000	130.000000	18.000000	
75%	1974.000000	1.000000	173.000000	77.00000	280.000000	140.000000	21.000000	
max	1993.000000	1.000000	180.000000	97.00000	568.000000	200.000000	29.000000	

In [31]:
```
data['Birth_Year'].hvplot.box()
```

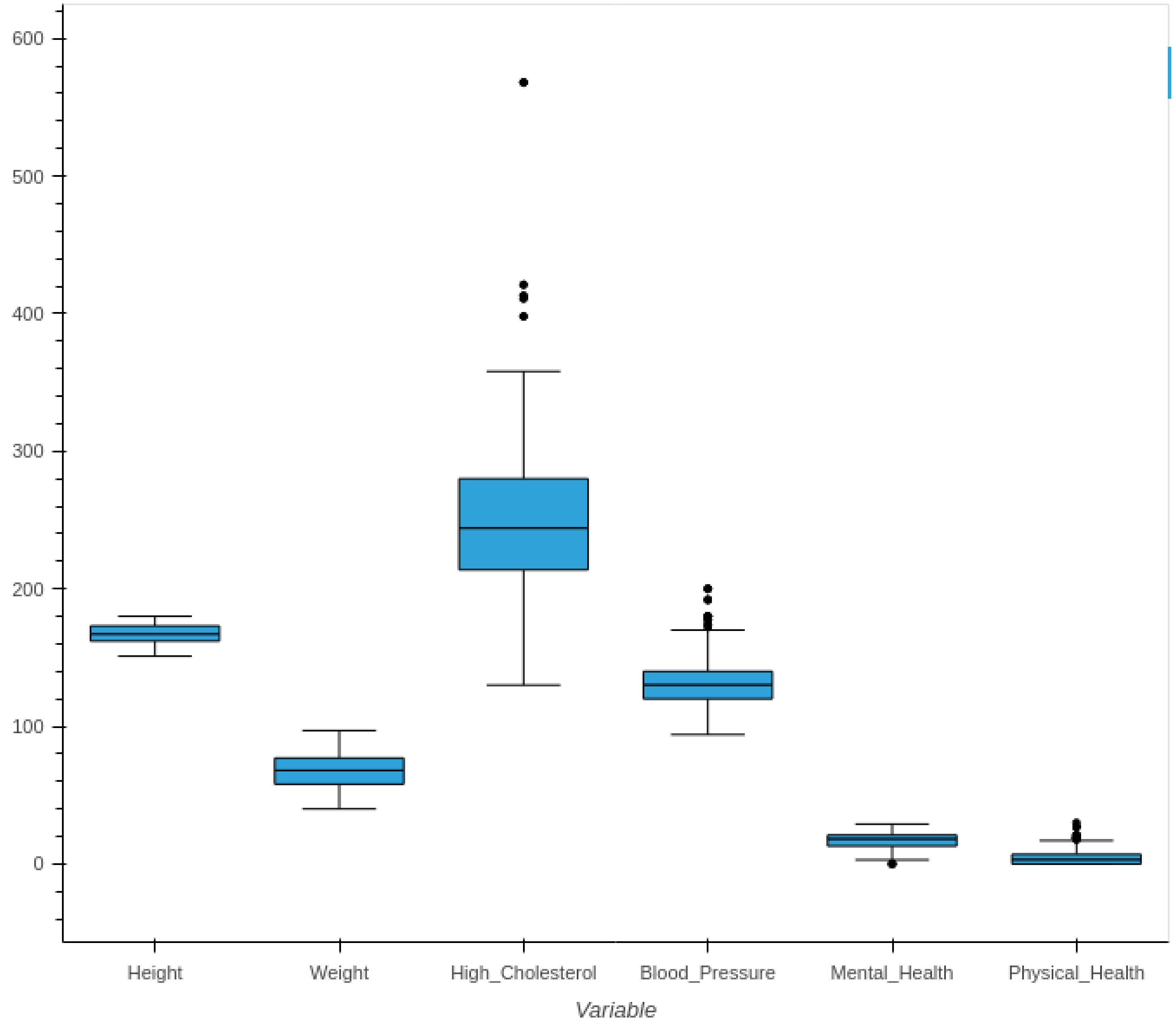
Out[31]:



Some outliers shows that some people was born before 1900, the feature can be parametrized in the lower end to have the value of the inferior whisker in the boxplot.

In [32]:
```
data.drop(columns=['PatientID', 'Birth_Year', 'Disease']).hvplot.box(height=600)
```

Out[32]:



In [ ]:

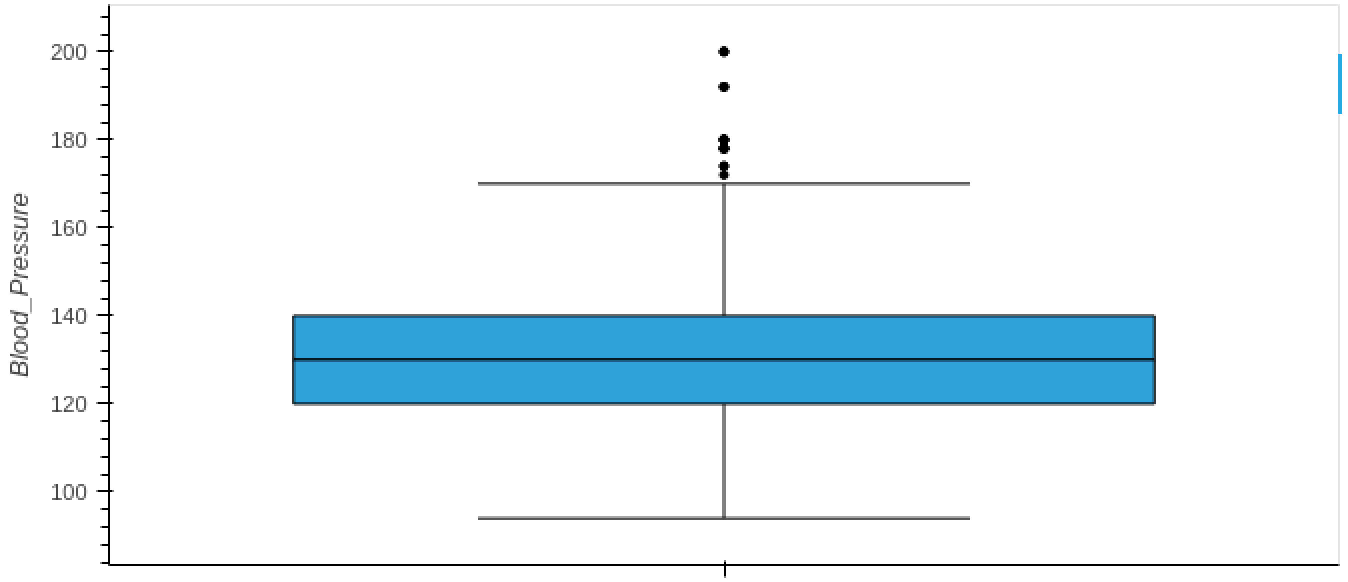
From all the numerical values, only `Height` and `Weight` have no outliers, further exploration will tell what to do with them.

`High_Cholesterol` has a very high outlier, while possible, the size of the data in that range is too small and the feature can be parametrized to a maximum cap.

In [33]:

```
data['Blood_Pressure'].hvplot.box()
```

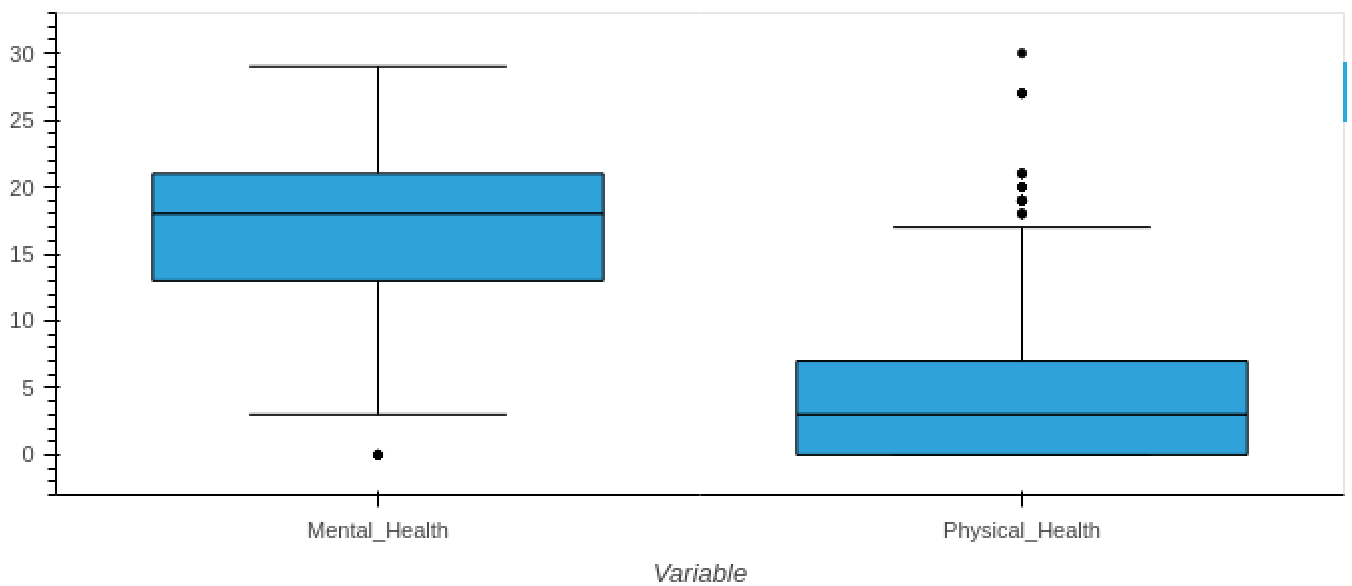
Out[33]:



Despite having some high values as outliers, this values are possible to exist and are not to far from the median. High blood pressure could represent a risk factor and it's effect in the model may be worth it to test it.

In [34]:
```
data[['Mental_Health', 'Physical_Health']].hvplot.box(hover=True)
```

Out[34]:



Missing Values

From the previous exploration, it can be checked that only `Education` has missing values. The test data has no missing values.

In [35]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 0 to 799
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   PatientID         800 non-null    object
 1   Name              800 non-null    object
 2   Birth_Year        800 non-null    int64 
 3   Region            800 non-null    object
 4   Education         787 non-null    object
 5   Disease           800 non-null    int64 
 6   Smoking_Habit     800 non-null    object
 7   Drinking_Habit    800 non-null    object
 8   Exercise          800 non-null    object
 9   Fruit_Habit       800 non-null    object
 10  Water_Habit       800 non-null    object
 11  Height            800 non-null    int64 
 12  Weight            800 non-null    int64 
 13  High_Cholesterol  800 non-null    int64 
 14  Blood_Pressure    800 non-null    int64 
 15  Mental_Health     800 non-null    int64 
 16  Physical_Health   800 non-null    int64 
 17  Checkup           800 non-null    object
 18  Diabetes          800 non-null    object
dtypes: int64(8), object(11)
memory usage: 125.0+ KB
```

In [36]:
```
data['Education'].isna().mean()
```

Out[36]:
```
0.01625
```

Missing values represent only 1.6 % and can be imputed with mean, median or KNN imputer. Nevertheless, `Education` seems to be not important for model prediction and dropping the entire feature is into consideration.

Data Cleaning

From all the numerical features with outliers only `High_Cholesterol` have some very far from it's median. any person with cholesterol around 400 have a very high cholesterol in blood, considering that the outliers are in a small frequency, it cabn be capped to 400.

In [ ]:
```

```

Data procesing

All the data procesing will be studied here, the actual processing will be done in custom transformers that can be fed into a pipeline.

That way we will be able to split the data for validation, fit the transformers in the training set and apply them in the validation and test set.

Feature Selection

Wih the visual inspection on the categorical features, it can be infere that the columns `['Name',` `Region', 'Education', 'Smoking_Habit', 'Water_Habit']` can be safely dropped. For numerical values, age (`Birth_Year`) seems that can be dropped too.

Below is the transformer to feed a pipeline in order to drop the features.

In [161…
```
class Dropper(BaseEstimator, TransformerMixin):
    def __init__(self, columns):
        self.columns = columns
    def fit(self, X, y=None):
        return self
    def transform(self, X, y=None):
        new_X = X.copy()
        new_X.drop(columns=self.columns, inplace=True)
        return new_X
```

In [162…
```
dropper = Dropper(to_drop)
dropped = dropper.fit_transform(data)
dropped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 0 to 799
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   PatientID         800 non-null    object
 1   Birth_Year        800 non-null    int64
 2   Disease           800 non-null    int64
 3   Drinking_Habit    800 non-null    object
 4   Exercise          800 non-null    object
 5   Fruit_Habit       800 non-null    object
 6   Height            800 non-null    int64
 7   Weight            800 non-null    int64
 8   High_Cholesterol  800 non-null    int64
 9   Blood_Pressure    800 non-null    int64
 10  Mental_Health     800 non-null    int64
 11  Physical_Health   800 non-null    int64
 12  Checkup           800 non-null    object
 13  Diabetes          800 non-null    object
dtypes: int64(8), object(6)
memory usage: 93.8+ KB
```

In [165…
```
dropper.transform(data_test).info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 225 entries, 0 to 224
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   PatientID         225 non-null    object
 1   Birth_Year        225 non-null    int64 
 2   Drinking_Habit    225 non-null    object
 3   Exercise          225 non-null    object
 4   Fruit_Habit       225 non-null    object
 5   Height            225 non-null    int64 
 6   Weight            225 non-null    int64 
 7   High_Cholesterol  225 non-null    int64 
 8   Blood_Pressure    225 non-null    int64 
 9   Mental_Health     225 non-null    int64 
 10  Physical_Health   225 non-null    int64 
 11  Checkup           225 non-null    object
 12  Diabetes          225 non-null    object
dtypes: int64(7), object(6)
memory usage: 24.6+ KB
```

Feature Engineering

Numerical features

According with the correlation matrix, Height and Weight do not explain too much the outcome, nevertheless, the BMI can be calculated using both features. For people who are considered obese (BMI greater than or equal to 30) or those who are overweight (BMI of 25 to 29.9) and have two or more risk factors, this is considered a risk factor for diabetees and cardio-respiratory diseases.

The BMI can be calculated through:

$$\frac{W}{h^2}$$

- W: weight in kilograms.
- h: height in meters.

In [37]:

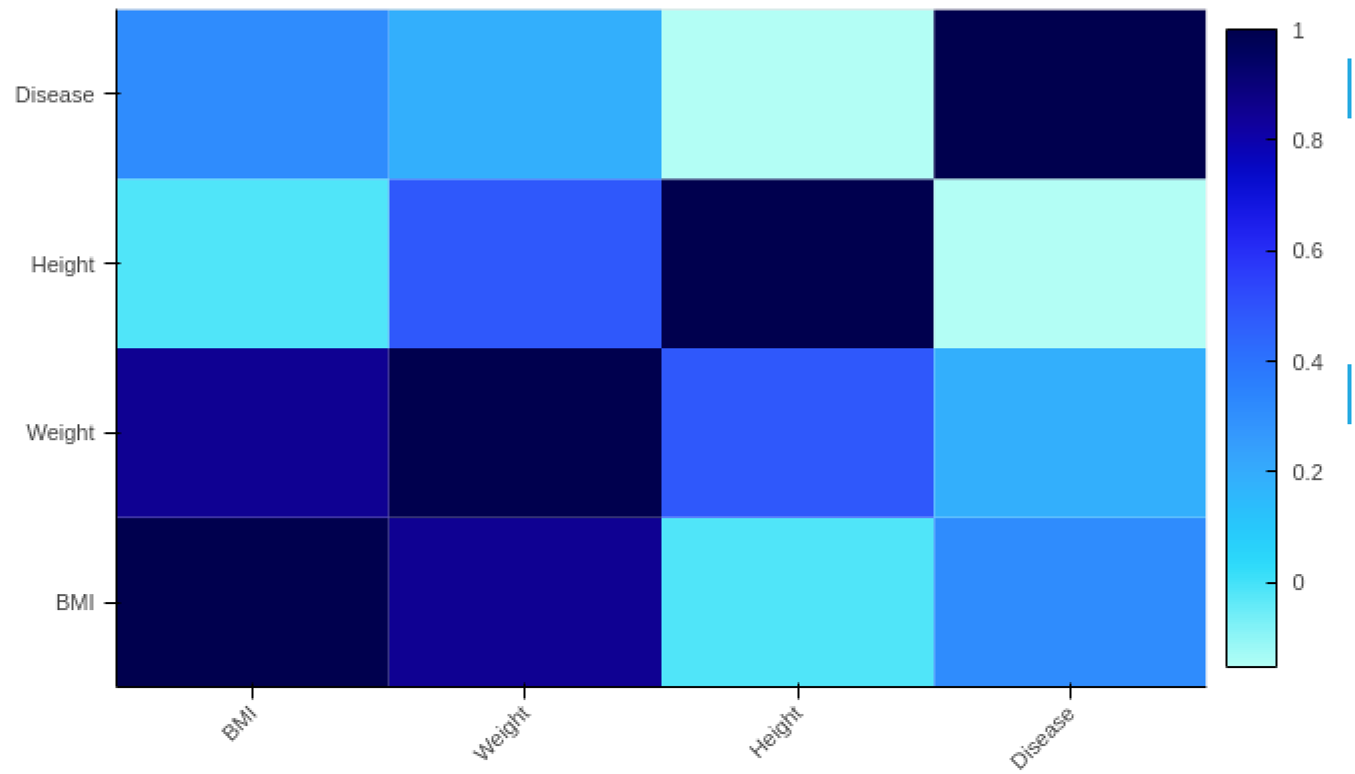
```
data_2 = data.copy()
numeric = data_2.select_dtypes('number')
numeric['BMI'] = numeric['Weight'] / (numeric['Height'] / 100) ** 2
# data_2.drop(columns=['Weight', 'Height'], inplace=True)
numeric[['BMI']].describe()
```

Out[37]:

	BMI
count	800.000000
mean	24.039237
std	3.658814
min	16.975309
25%	20.830526
50%	24.382545
75%	27.147402
max	30.119402

In [38]:
```python
numeric.corr(method='spearman').loc[['BMI', 'Weight', 'Height', 'Disease'],
                                     ['BMI', 'Weight', 'Height', 'Disease']].hvplot.heatma
```

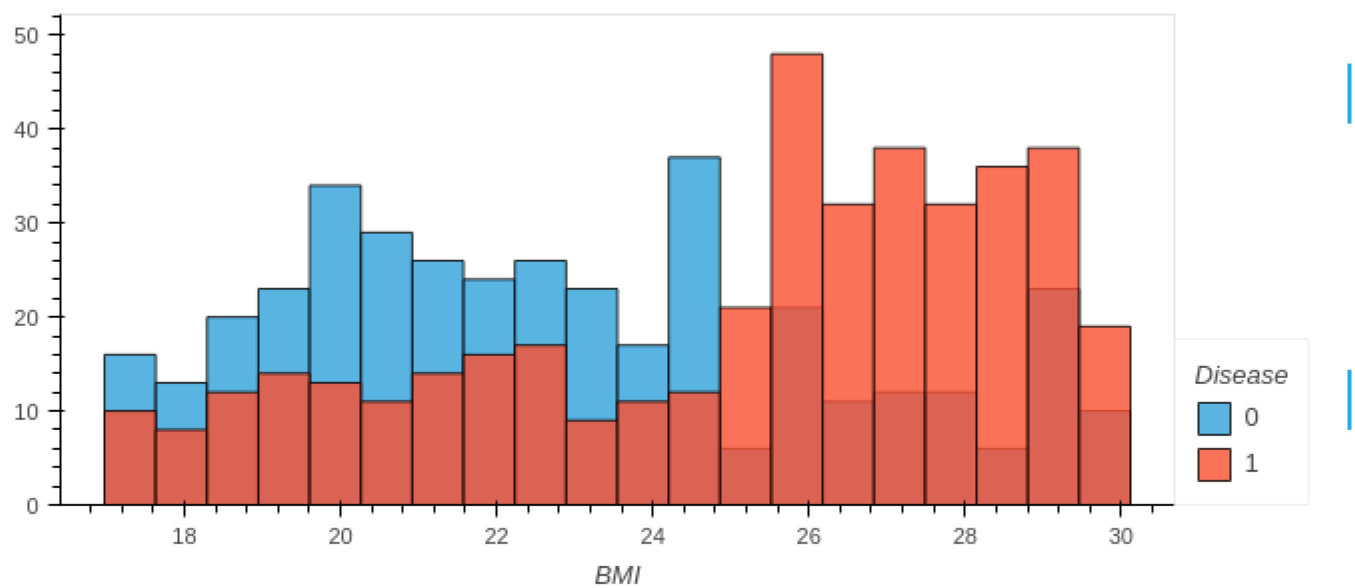
Out[38]:



Hovering on the heatmap reveals the correlation values. The new feature `BMI` has almost twice the correlation than the best of `Height` and `Weight`. These two features can be safely drop in favor of the `BMI` which serves as a better predictor to the model and holds the information of `Height` and `Weight`.

In [39]:
```python
numeric.hvplot.hist(y='BMI', by='Disease', alpha=0.8)
```

Out[39]:



Values from infected and not infected seems to be from different distributions and people with higher `BMI` are more prone to be infected.

Transformer for sckit-learn

In [40]:
```python
class BMIConstructor(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self
    def transform(self, X, y=None):
        new_X = X.copy()
        new_X['BMI'] = new_X['Weight'] / (new_X['Height'] / 100) ** 2
        new_X.drop(columns=['Weight', 'Height'], inplace=True)
        return new_X
```

In [45]:
```python
# test transformer

bmi = BMIConstructor()
transformed = bmi.fit_transform(data)
transformed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 0 to 799
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   PatientID         800 non-null    object 
 1   Name              800 non-null    object 
 2   Birth_Year        800 non-null    int64  
 3   Region            800 non-null    object 
 4   Education         787 non-null    object 
 5   Disease           800 non-null    int64  
 6   Smoking_Habit     800 non-null    object 
 7   Drinking_Habit    800 non-null    object 
 8   Exercise          800 non-null    object 
 9   Fruit_Habit       800 non-null    object 
 10  Water_Habit       800 non-null    object 
 11  High_Cholesterol  800 non-null    int64  
 12  Blood_Pressure    800 non-null    int64  
 13  Mental_Health     800 non-null    int64  
 14  Physical_Health   800 non-null    int64  
 15  Checkup           800 non-null    object 
 16  Diabetes          800 non-null    object 
 17  BMI               800 non-null    float64
dtypes: float64(1), int64(6), object(11)
memory usage: 118.8+ KB
```

In [46]:
```python
# test data transformation
test_transformed = bmi.transform(data_test)
test_transformed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 225 entries, 0 to 224
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   PatientID         225 non-null    object 
 1   Name              225 non-null    object 
 2   Birth_Year        225 non-null    int64  
 3   Region            225 non-null    object 
 4   Education         225 non-null    object 
 5   Smoking_Habit     225 non-null    object 
 6   Drinking_Habit    225 non-null    object 
 7   Exercise          225 non-null    object 
 8   Fruit_Habit       225 non-null    object 
 9   Water_Habit       225 non-null    object 
 10  High_Cholesterol  225 non-null    int64  
 11  Blood_Pressure    225 non-null    int64  
 12  Mental_Health     225 non-null    int64  
 13  Physical_Health   225 non-null    int64  
 14  Checkup           225 non-null    object 
 15  Diabetes          225 non-null    object 
 16  BMI               225 non-null    float64
dtypes: float64(1), int64(5), object(11)
memory usage: 31.6+ KB
```

Categorical features

Most categorical features have more than two values. To avoid the curse of dimensionality, some of the values can be combined to reduce the number of columns once they are dummified.

One examples of this is `Diabetes` with values:

`'Neither I nor my immediate family have diabetes.'`

`'I have/had pregnancy diabetes or borderline diabetes',`

`'I do have diabetes',`

`"I don't have diabetes, but I have direct family members who have diabetes."`

Values can be combine into two: Diabetes and non-Diabetes, turning it in a binary feature.

Other feature that can have this kind of treatment are

`> Drinking_Habit`

`> Fruit_Habit`

`> Checkup`

Drinking Habit

For this the two options that consume less drinks or no drinks at all can be combined into one.

In [48]:
```
cat['Drinking_Habit'].unique()
```

Out[48]: array(['I usually consume alcohol every day',
       'I consider myself a social drinker',
       'I do not consume any type of alcohol'], dtype=object)

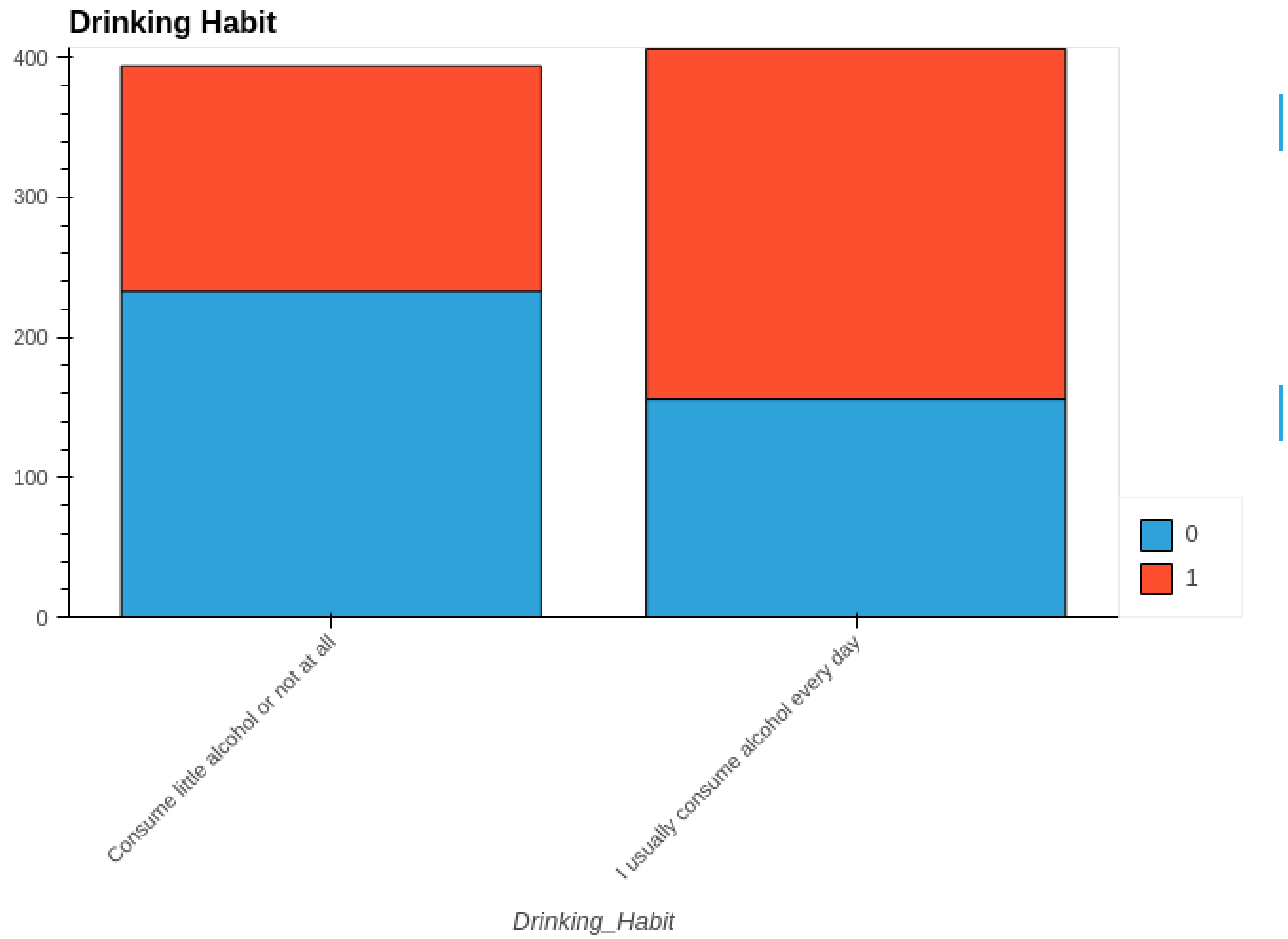
In [49]:
```
drinker, soc_drinker, no_drinker = cat['Drinking_Habit'].unique()
new_drinker = 'Consume little alcohol or not at all'
cat['Drinking_Habit'][(cat['Drinking_Habit'] == soc_drinker) | (cat['Drinking_Habit'] ==

cat['Drinking_Habit'].unique()
```

Out[49]: array(['I usually consume alcohol every day',
       'Consume little alcohol or not at all'], dtype=object)

In [50]:
```
countplot(cat, 'Drinking_Habit', 'Disease', 500)
```

Out[50]:



Fruit Habit

In [51]:
```
cat['Fruit_Habit'].unique()
```

Out[51]: array(['Less than 1. I do not consume fruits every day.',
       '1 to 2 pieces of fruit in average',
       '3 to 4 pieces of fruit in average',
       '5 to 6 pieces of fruit in average',
       'More than six pieces of fruit'], dtype=object)

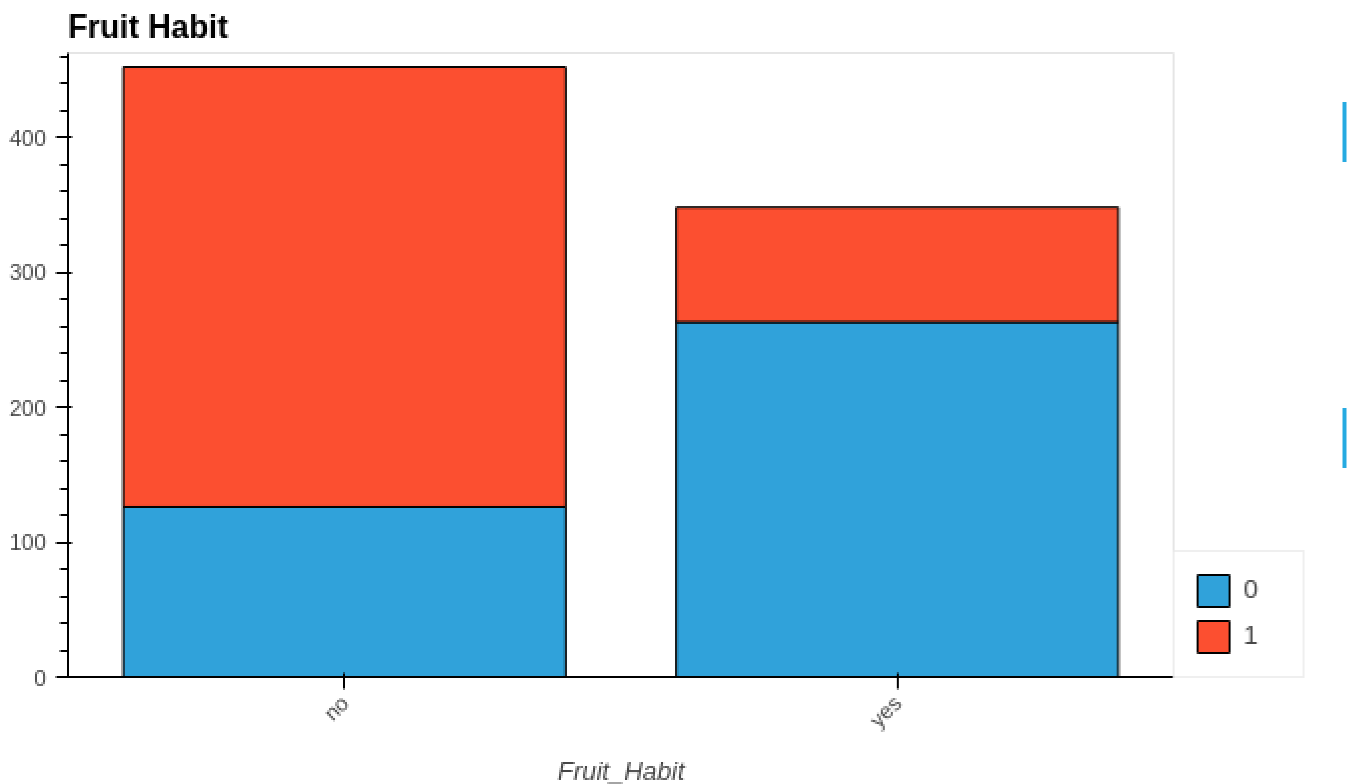
In [52]:
```
less_1, fruit_12, fruit_34, fruit_56, more_6 = cat['Fruit_Habit'].unique()
column = 'Fruit_Habit'
cat[column][(cat[column] == fruit_12) | (cat[column] == fruit_34)\
            | (cat[column] == fruit_56) | (cat[column] == more_6)] = 'yes'
cat[column][cat[column] == less_1] = 'no'
```

```
cat['Fruit_Habit'].unique()
```

Out[52]:
```
array(['no', 'yes'], dtype=object)
```

In [53]:
```
countplot(cat, 'Fruit_Habit', 'Disease')
```

Out[53]:



Checkup

Hard assumptions are required. for instance, the `Not Sure` value could mean anything and probably a mix of all the other options. People in the `Not Sure` category are less likely to be infected and can be grouped in `Less than 3 years`.

In [54]:
```
cat['Checkup'].unique()
```

Out[54]:
```
array(['More than 3 years', 'Not sure',
       'Less than 3 years but more than 1 year', 'Less than three months'],
      dtype=object)
```

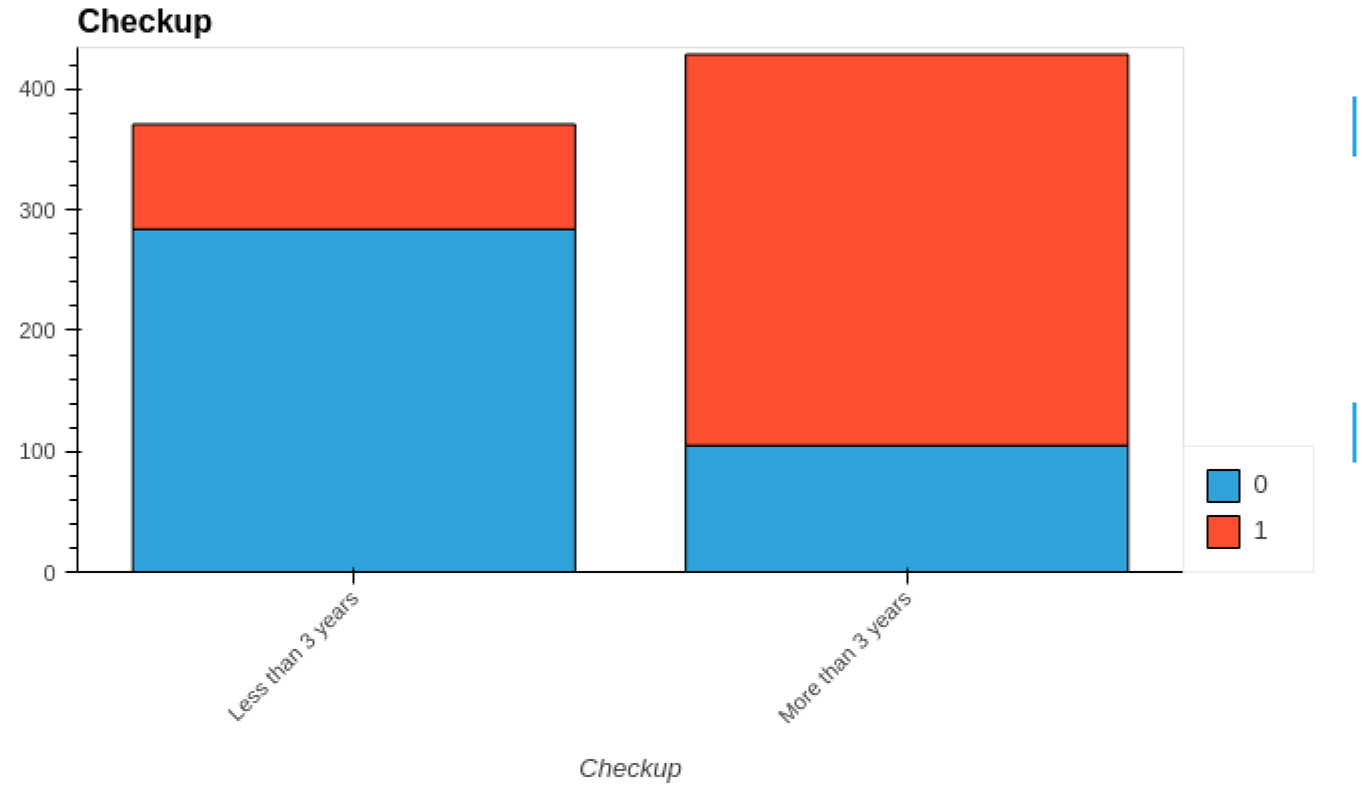
In [55]:
```
checkup_1, checkup_2, checkup_3, checkup_4 = cat['Checkup'].unique()
cat['Checkup'][(cat['Checkup'] == checkup_2) | (cat['Checkup'] == checkup_3) |\
               (cat['Checkup'] == checkup_4)] = 'Less than 3 years'

cat['Checkup'].unique()
```

Out[55]:
```
array(['More than 3 years', 'Less than 3 years'], dtype=object)
```

In [56]:
```
countplot(cat, 'Checkup', 'Disease')
```

Out[56]:



Diabetes

In [57]:
```
cat['Diabetes'].unique()
```

Out[57]:
```
array(['Neither I nor my immediate family have diabetes.',
       'I have/had pregnancy diabetes or borderline diabetes',
       'I do have diabetes',
       "I don't have diabetes, but I have direct family members who have diabetes."],
      dtype=object)
```

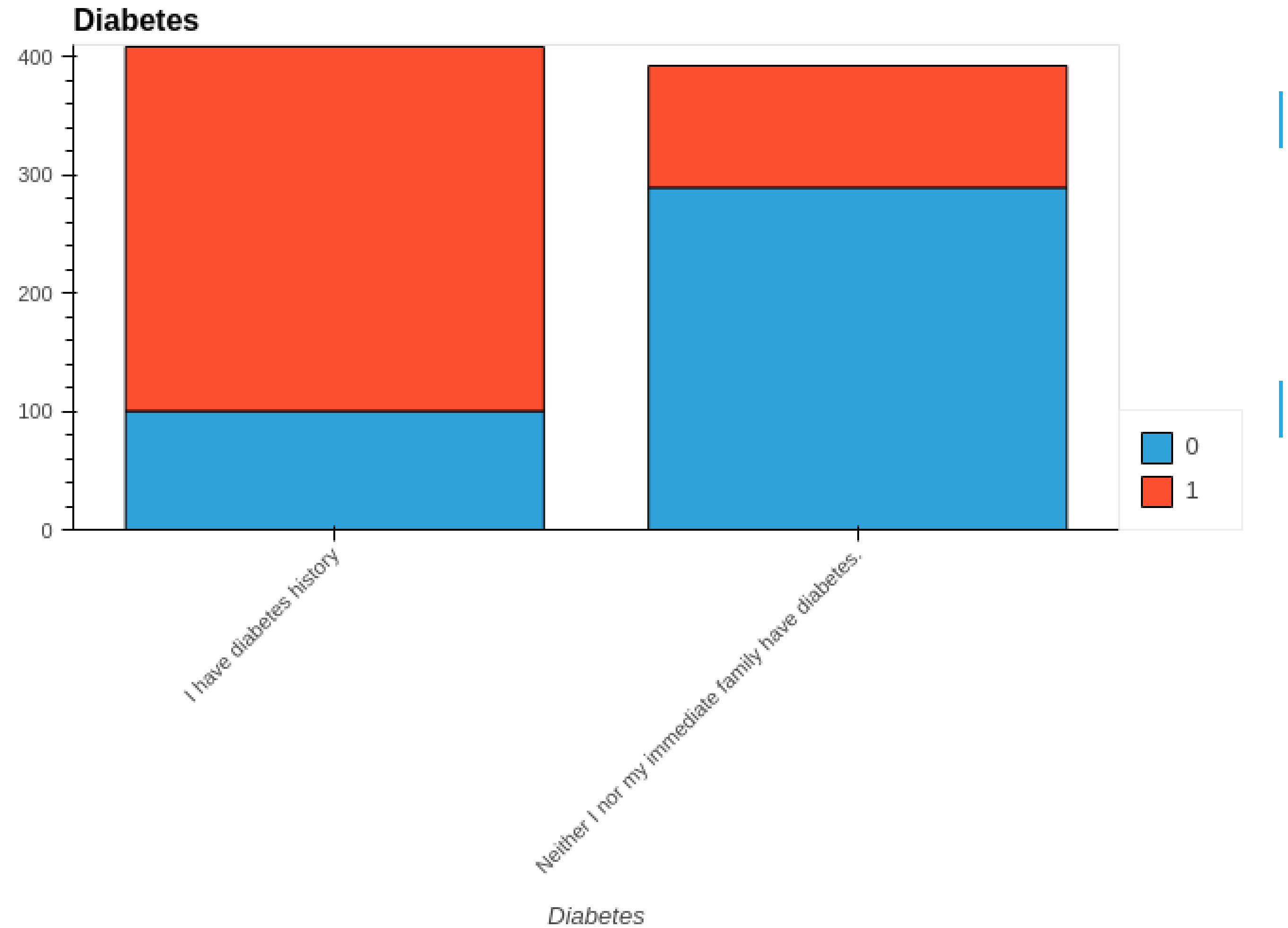
In [58]:
```
diabetes_1, diabetes_2, diabetes_3, diabetes_4 = cat['Diabetes'].unique()
cat['Diabetes'][(cat['Diabetes'] == diabetes_2) | (cat['Diabetes'] == diabetes_3) |\
               (cat['Diabetes'] == diabetes_4)] = 'I have diabetes history'

cat['Diabetes'].unique()
```

Out[58]:
```
array(['Neither I nor my immediate family have diabetes.',
       'I have diabetes history'], dtype=object)
```

In [59]:
```
countplot(cat, 'Diabetes', 'Disease', 500)
```

Out[59]:



Data Encoding

One Hot encoding

In [ ]:

In [ ]:

Scaling

In [ ]:

Train-Test Split

In [ ]:

Modeling

Model Selection

In [ ]:

In [ ]:

In [ ]:

Model Implementation

In [ ]:

Evaluation

In [ ]:

In [ ]:

In [ ]:

Conclusion

In [ ]:

In [ ]:

In [ ]: