Heart Disease using Liner Regression

Team Members

Sama Hesham Sayed
Rana Hamdy Abdelkader
Riham Mohsen Sayed
Hussein Ibrahim Hassan
Marwan Ahmed Mohamed
Ahmed Hamdy Abdel Sattar

Introduction:

- Linear regression is a fundamental statistical technique used to model and analyze the relationship between two or more variables. It is a powerful tool for predicting or estimating a continuous target variable based on one or more independent variables. Linear regression assumes a linear relationship between the input variables and the target variable, making it widely applicable in various fields such as economics, finance, social sciences, and machine learning.
- In summary, linear regression is a widely used statistical technique for modeling and predicting the relationship between variables. By estimating the coefficients of the regression line, it enables us to make informed predictions, understand the impact of independent variables on the dependent variable, and uncover valuable insights from data.
- heart disease is one of the leading causes of death for people of most races in the US (African Americans, American Indians and Alaska Natives, and white people). 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. Detecting and preventing the factors that have the greatest impact on heart disease is very important in healthcare. Computational developments, in turn, allow the application of machine learning methods to detect "patterns" from the data that can predict a patient's condition.

Dataset:

- Our dataset have some attributes to predict the heart disease
- Explanation of the variables of the dataset:
  - BMI : Body Mass Index (BMI).
  - . Smoking : Have you smoked at least 100 cigarettes in your entire life? (The answer Yes or No).
  - Stroke : (Ever told) (you had) a stroke?
  - Sex : Are you male or female?
  - AgeCategory: Fourteen-level age category.
  - Race : Imputed race/ethnicity value.
  - GenHealth : Would you say that in general your health is…
  - SleepTime : On average, how many hours of sleep do you get in a 24-hour period?

- Asthma : (Ever told) (you had) asthma?
- SkinCancer : (Ever told) (you had) skin cancer?
- PhysicalActivity : Adults who reported doing physical activity or exercise during the past 30 days other than their regular job.
- DiffWalking : Do you have serious difficulty walking or climbing stairs?
- MentalHealth : Thinking about your mental health, for how many days during the past 30 days was your mental health not good? (0-30 days).
- PhysicalHealth : Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good? (0-30 days).
- AlcoholDrinking : Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week

The source of the data Kaggle.com link :

("Heart Disease Prediction | Kaggle")

Data preprocessing:

Data preprocessing involves assigning concatenating two tables and Encode the training data

```
transformer = make_column_transformer(
    (OneHotEncoder(sparse=False), ['AgeCategory', 'Race', 'GenHealth']),
    remainder='passthrough')

# Encode training data
transformed_train = transformer.fit_transform(X_train)
transformed_train_data = pd.DataFrame(transformed_train, columns=transformer.get_feature_names_out())

# Concat the two tables
transformed_train_data.reset_index(drop=True, inplace=True)
X_train.reset_index(drop=True, inplace=True)
X_train = pd.concat([transformed_train_data, X_train], axis=1)
```

Removing columns and encoding the test data:

```
# Remove old columns
X_train.drop(['AgeCategory', 'Race', 'GenHealth'], axis=1, inplace=True)

# Encode test data
transformed_test = transformer.transform(X_test)
transformed_test_data = pd.DataFrame(transformed_test, columns=transformer.get_feature_names_out())
```

Reading the dataset the csv file for on the model :

```
df = pd.read_csv("/content/drive/MyDrive/heartdisease/heart_2020_cleaned2.csv")
df.head()
```

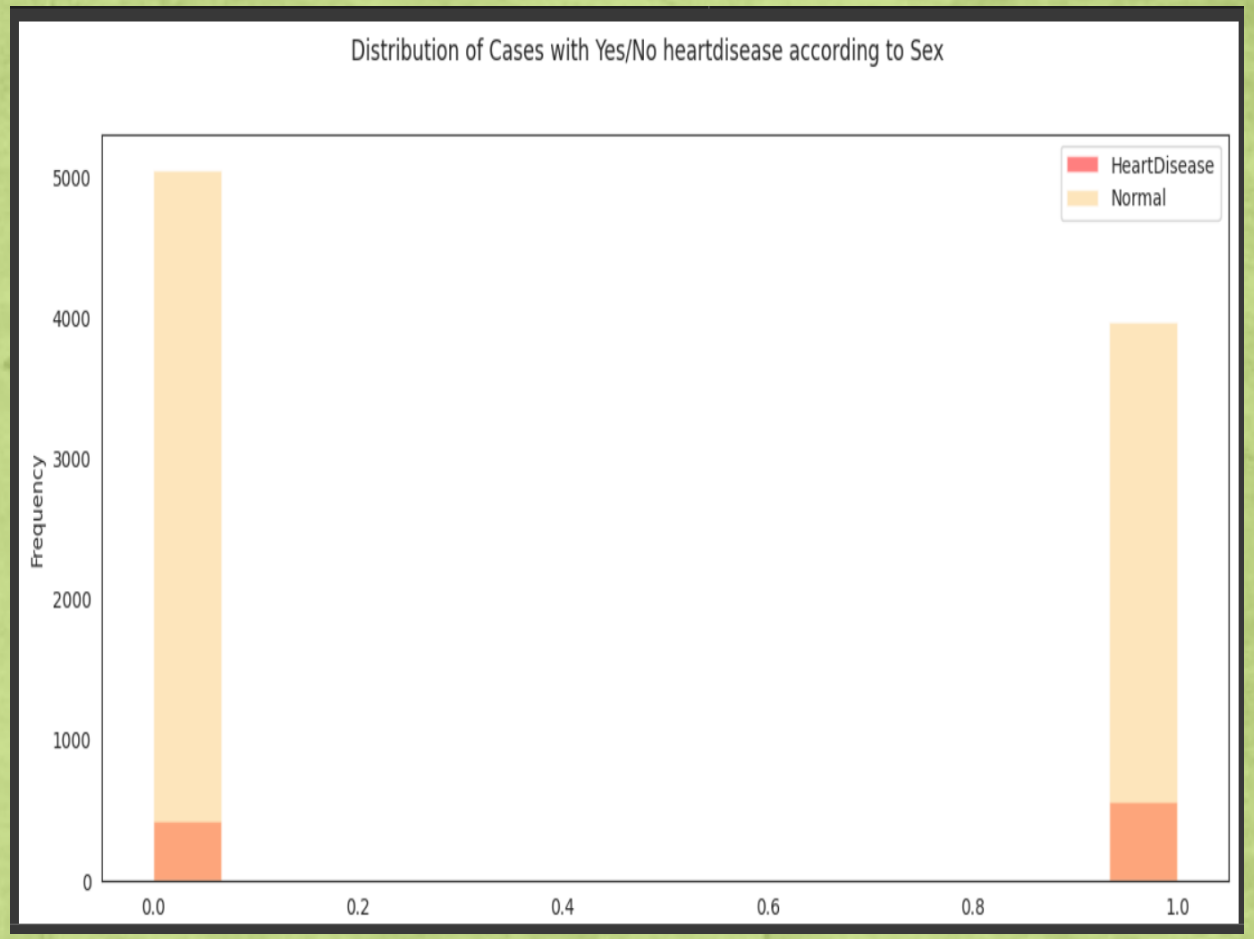
Split Dataset for Training and Testing

```
[ ] # Select Features
    features = df.drop(columns=['HeartDisease'], axis=1)

    # Select Target
    target = df['HeartDisease']

    # Set Training and Testing Data
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(features, target, shuffle=True, test_size=.2, random_state=44)

    print('Shape of training feature:', X_train.shape)
    print('Shape of testing feature:', X_test.shape)
    print('Shape of training label:', y_train.shape)
    print('Shape of training label:', y_test.shape)
```

Data Visualization

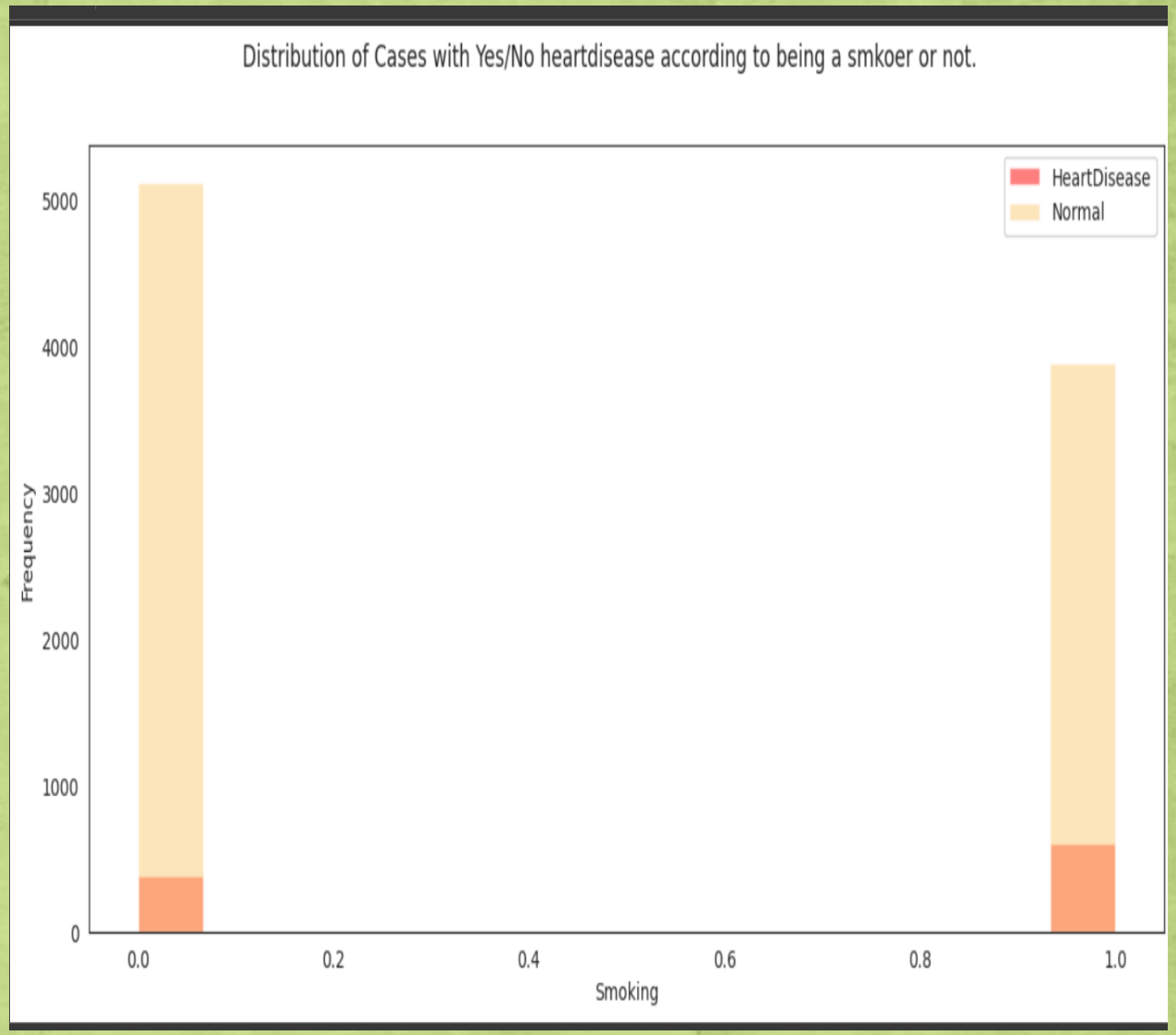
```
[ ] df =  df[df.columns].replace({'Yes':1, 'No':0, 'Male':1,'Female':0,'No, borderline diabetes':'0','Yes (during pregnancy)':'1' })
    df['Diabetic'] = df['Diabetic'].astype(int)
```

```
fig, ax = plt.subplots(figsize = (13,6))

ax.hist(df[df["HeartDisease"]==1]["Sex"], bins=15, alpha=0.5, color="red", label="HeartDisease")
ax.hist(df[df["HeartDisease"]==0]["Sex"], bins=15, alpha=0.5, color="#fccc79", label="Normal")

ax.set_xlabel("Sex")
ax.set_ylabel("Frequency")

fig.suptitle("Distribution of Cases with Yes/No heartdisease according to Sex")

ax.legend();
```



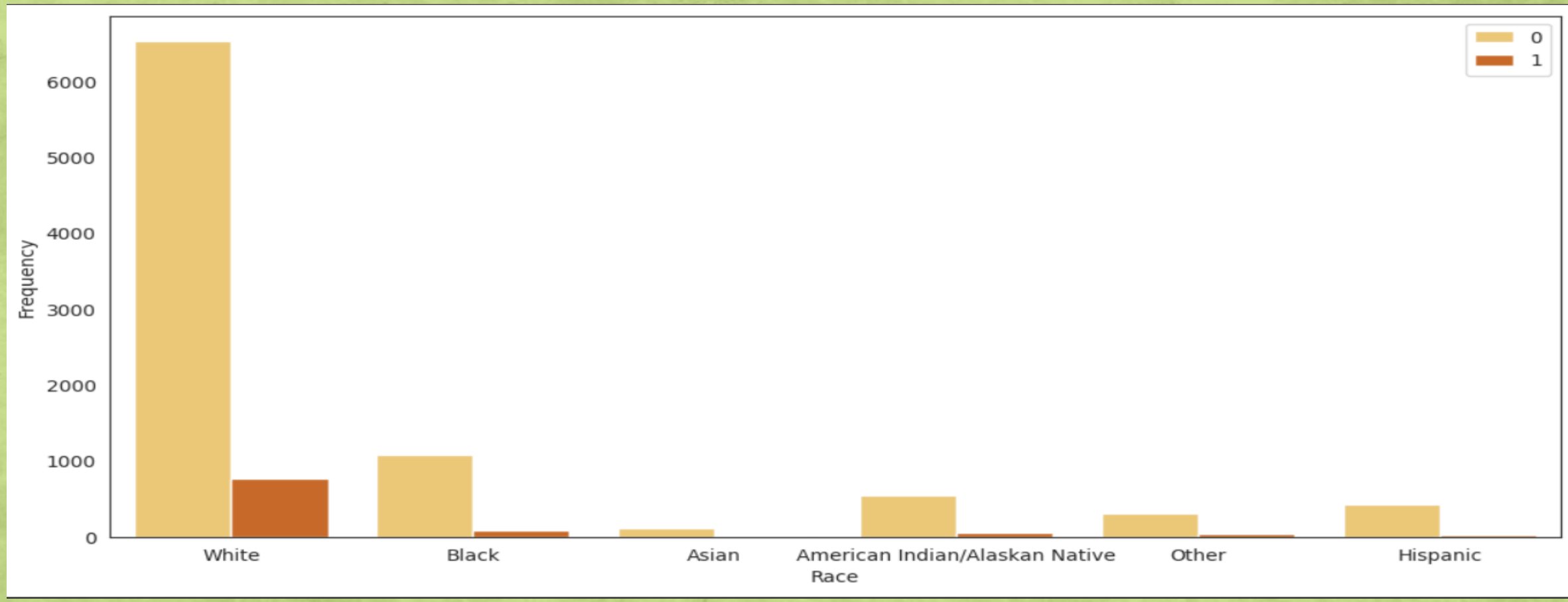
If the patient smokes or not
And the distribution of smoking people
:

```
fig, ax = plt.subplots(figsize = (13,6))

ax.hist(df[df["HeartDisease"]==1]["Smoking"], bins=15, alpha=0.5, color="red", label="HeartDisease")
ax.hist(df[df["HeartDisease"]==0]["Smoking"], bins=15, alpha=0.5, color="#fccc79", label="Normal")

ax.set_xlabel("Smoking")
ax.set_ylabel("Frequency")

fig.suptitle("Distribution of Cases with Yes/No heartdisease according to being a smkoer or not.")

ax.legend();
```



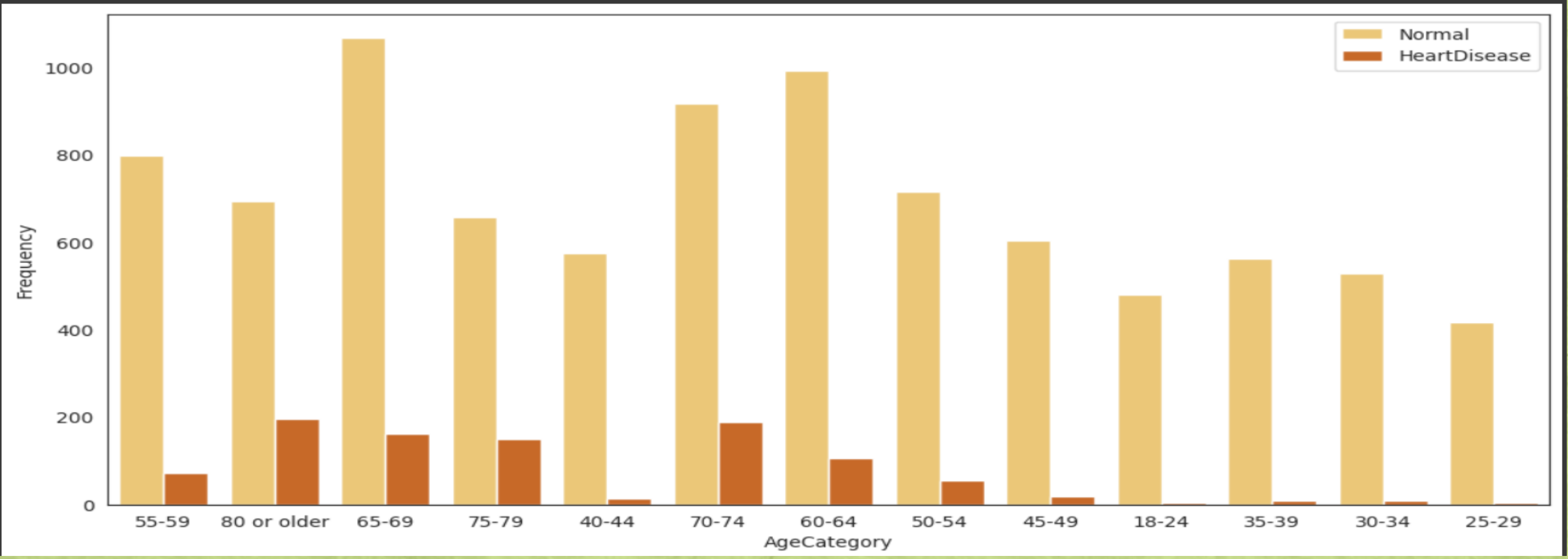
The Nationalities of the Dataset :

```
[ ]  plt.figure(figsize = (13,6))
     sns.countplot( x= df['Race'], hue = 'HeartDisease', data = df, palette = 'YlOrBr')
     plt.xlabel('Race')
     plt.legend()
     plt.ylabel('Frequency')
     plt.show()
```



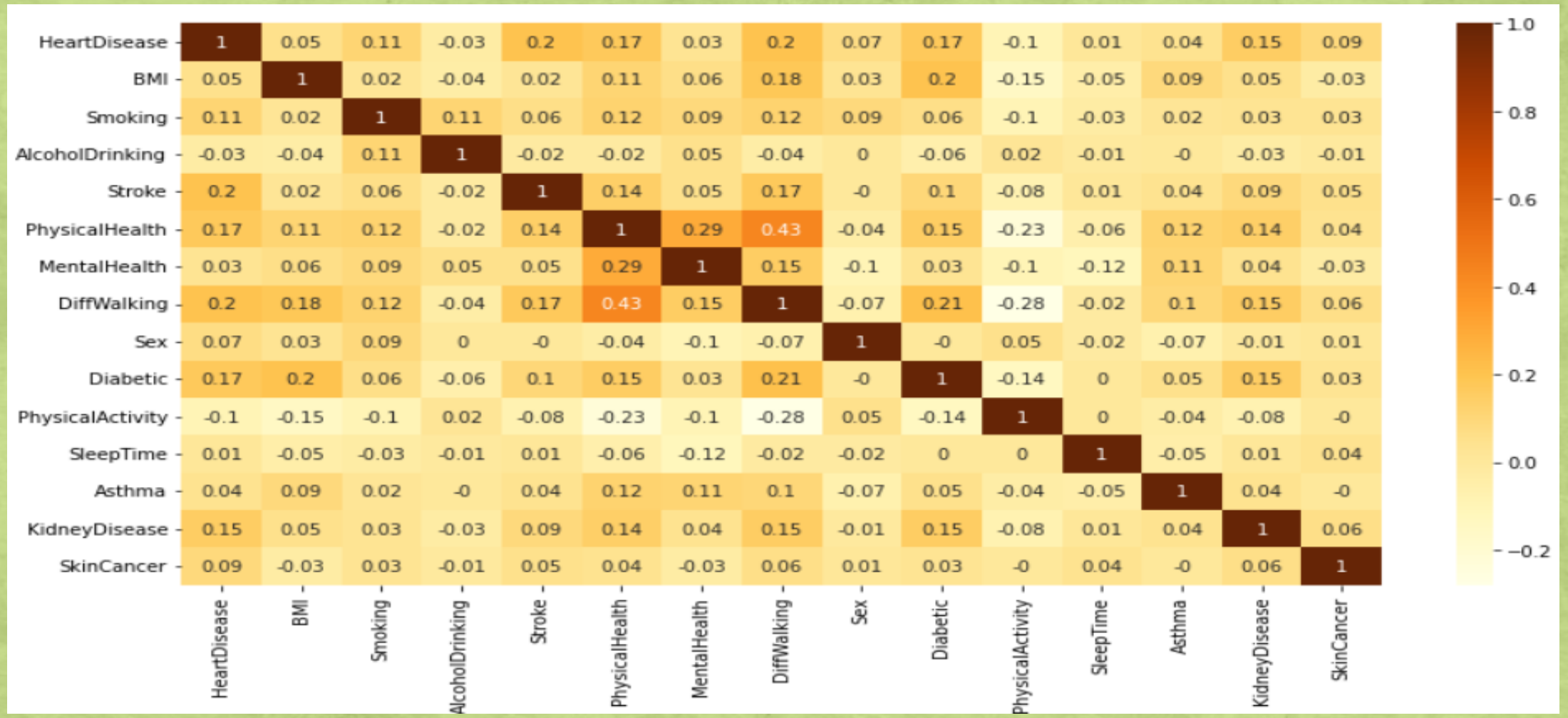
AGES :

```
plt.figure(figsize = (13,6))
sns.countplot(x = df['AgeCategory'], hue = 'HeartDisease', data = df, palette = 'YlOrBr')
fig.suptitle("Distribution of Cases with Yes/No hartdisease according to AgeCategory")
plt.xlabel('AgeCategory')
plt.legend(['Normal','HeartDisease'])
plt.ylabel('Frequency')
plt.show()
```

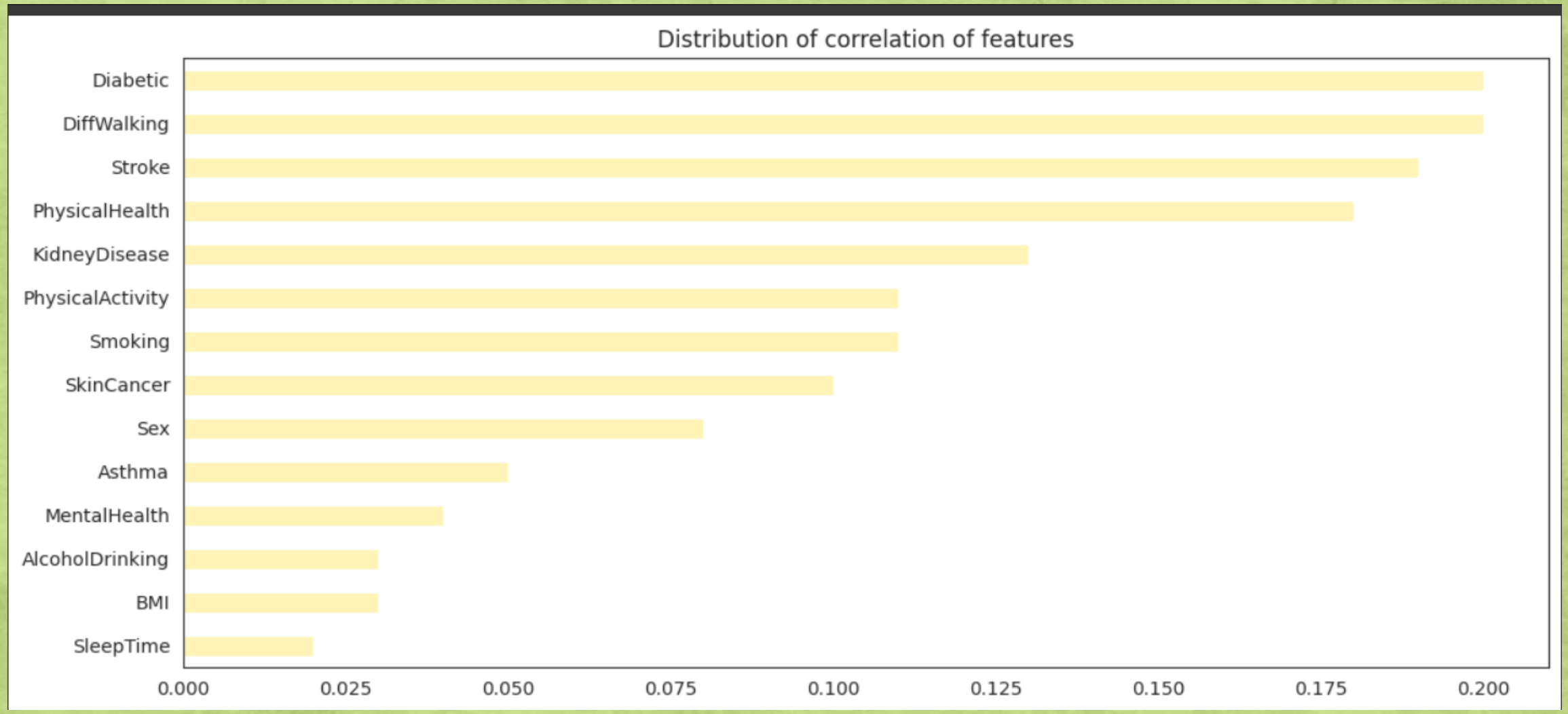


Visualization Of Numerical Features :

```
correlation = df.corr().round(2)
plt.figure(figsize = (14,7))
sns.heatmap(correlation, annot = True, cmap = 'YlOrBr')
```

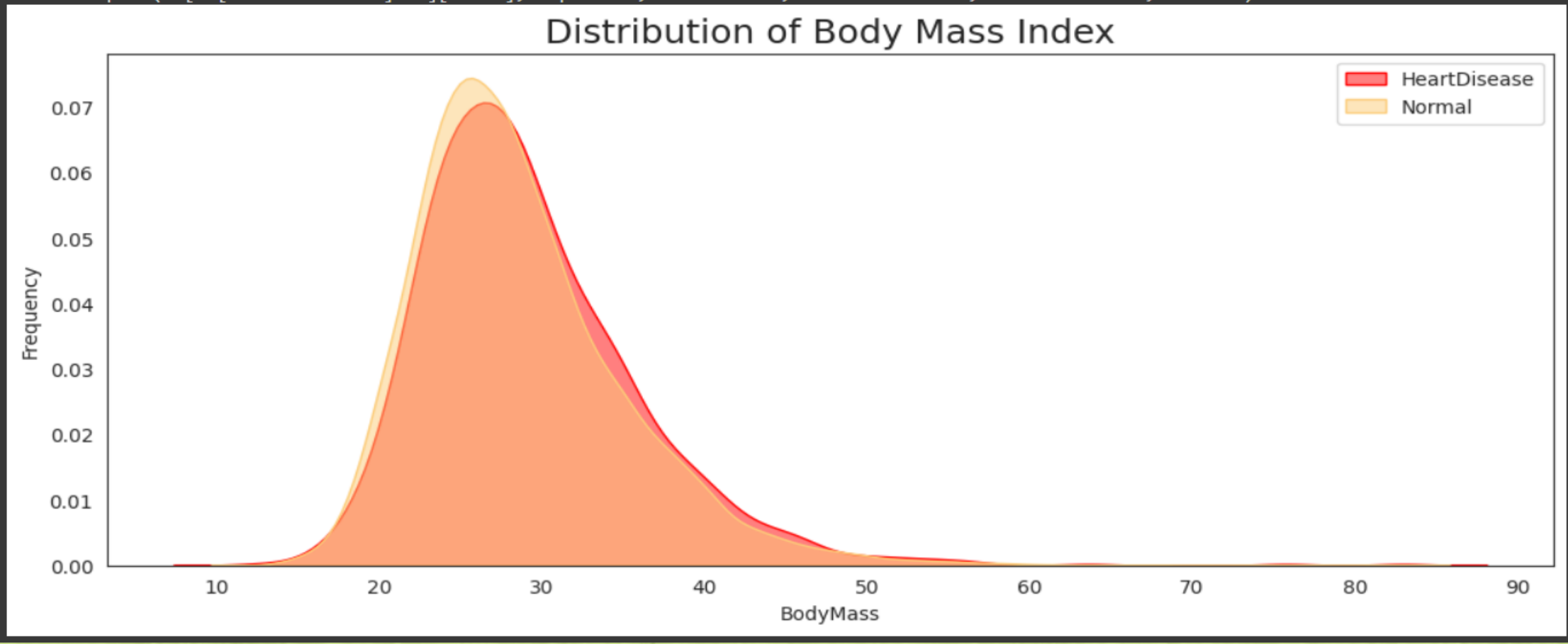


```
sns.set_style('white')
sns.set_palette('YlOrBr')
plt.figure(figsize = (13,6))
plt.title('Distribution of correlation of features')
abs(correlation['HeartDisease']).sort_values()[:-1].plot.barh()
plt.show()
```

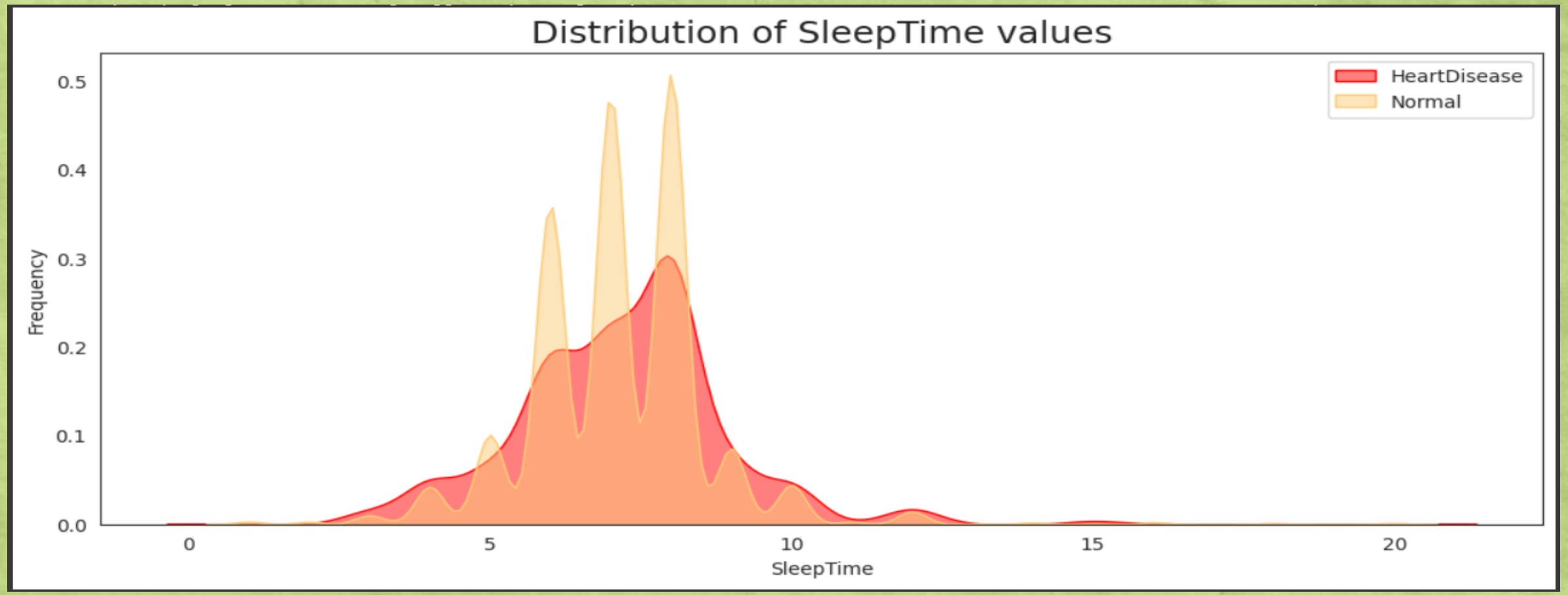


```
[ ]  fig, ax = plt.subplots(figsize = (13,5))
     sns.kdeplot(df[df["HeartDisease"]==1]["BMI"], alpha=0.5,shade = True, color="red", label="HeartDisease", ax = ax)
     sns.kdeplot(df[df["HeartDisease"]==0]["BMI"], alpha=0.5,shade = True, color="#fccc79", label="Normal", ax = ax)
     plt.title('Distribution of Body Mass Index', fontsize = 18)
     ax.set_xlabel("BodyMass")
     ax.set_ylabel("Frequency")
     ax.legend();
     plt.show()
```

```
sns.kdeplot(df[df["HeartDisease"]==0]["BMI"], alpha=0.5,shade = True, color="#fccc79", label="Normal", ax = ax)
```



```
fig, ax = plt.subplots(figsize = (13,5))
sns.kdeplot(df[df["HeartDisease"]==1]["SleepTime"], alpha=0.5,shade = True, color="red", label="HeartDisease", ax = ax)
sns.kdeplot(df[df["HeartDisease"]==0]["SleepTime"], alpha=0.5,shade = True, color="#fccc79", label="Normal", ax = ax)
plt.title('Distribution of SleepTime values', fontsize = 18)
ax.set_xlabel("SleepTime")
ax.set_ylabel("Frequency")
ax.legend();
plt.show()
```



Defining and compiling the training model :

- **Model Definition and Training :** Import the necessary libraries for linear regression, such as scikit-learn in Python and Define an instance of the linear regression model .
- **Model compilation :** Since linear regression is a simple and straightforward algorithm, it does not require explicit compilation like some other models.
- **Model Evaluation :** Use the trained linear regression model to make predictions on the testing data.
- **Model Deployment :** Once you are satisfied with the performance of the trained linear regression model, you can deploy it to make predictions on new, unseen data.

Building Model

Building the linear Regression Model (Using SGD):

```python
import numpy as np
from sklearn import metrics

# Define learning rate and number of iterations
learning_rate = 0.01
num_iterations_list = [500, 1000, 1500, 2000]  # Different values of num_iterations to try

best_mse = float('inf')
best_num_iterations = None

# Add bias term to the feature matrix
X_train_bias = np.c_[np.ones(X_train.shape[0]), X_train]

# Iterate over different values of num_iterations
for num_iterations in num_iterations_list:
    # Initialize model parameters
    num_features = X_train_bias.shape[1]
    theta = np.zeros(num_features)

    # Perform SGD
    for iteration in range(num_iterations):
        # Randomly shuffle the training data
        indices = np.random.permutation(X_train.shape[0])
        X_train_shuffled = X_train_bias[indices]
        y_train_shuffled = y_train.iloc[indices]

        # Iterate over each training example
        for i in range(X_train.shape[0]):
            # Compute the prediction and error
```

The complete of building model then calculate the statistical measures

```
        # Calculate metrics: MSE, MAE, R^2
        mse = metrics.mean_squared_error(y_test, y_pred)
        mae = metrics.mean_absolute_error(y_test, y_pred)
        r2 = metrics.r2_score(y_test, y_pred)

        # Print metrics for each iteration number in num_iterations_list
        if iteration+1 == num_iterations:
            print(f"Iteration: {num_iterations}")
            print("MSE:", mse)
            print("MAE:", mae)
            print("R^2:", r2)
            print()

    # Check if current model performs better
    if mse < best_mse:
        best_mse = mse
        best_num_iterations = num_iterations

# Print the best performing model
print("Best num_iterations:", best_num_iterations)
print("Best MSE:", best_mse)
```

The Statistical Measures :

```
# Print the best performing model
print("Best num_iterations:", best_num_iterations)
print("Best MSE:", best_mse)
```

```
Iteration: 500
MSE: 0.111207335111846 77
MAE: 0.22418838631066 11
R^2: -0.12696049931441067

Iteration: 1000
MSE: 0.1143950867761727
MAE: 0.23926180784308046
R^2: -0.15926475517762317

Iteration: 1500
MSE: 0.14265069901525926
MAE: 0.278990411669943 87
R^2: -0.4456034112147387

Iteration: 2000
MSE: 0.106092642468553 14
MAE: 0.189527274017117 25
R^2: -0.07512887715271854

Best num_iterations: 2000
Best MSE: 0.106092642468553 14
```

Thank You