## Implement KNN Algorithm

## https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset?resource=download

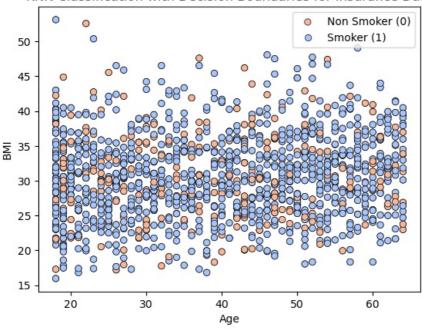
```
In [19]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
In [21]: from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler, LabelEncoder
         from sklearn.neighbors import KNeighborsClassifier # Changed KNeigborsClassifier to KNeighborsClassifier
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score, classification_report
         import seaborn as sns
In [25]: data = pd.read_csv("C:/Users/Acer/Downloads/insurance.csv")
In [27]: data.columns
Out[27]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
In [29]: data.shape
Out[29]: (1338, 7)
In [36]: data.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 1338 entries, 0 to 1337
        Data columns (total 7 columns):
         # Column Non-Null Count Dtype
        ---
            -----
                      -----
         0 age
                    1338 non-null int64
                 1338 non-null object
1338 non-null float6
         1 sex
                                      float64
         3 children 1338 non-null int64
         4 smoker 1338 non-null object
         5 region 1338 non-null object
6 charges 1338 non-null float6
                                       float64
        dtypes: float64(2), int64(2), object(3)
        memory usage: 73.3+ KB
In [38]: # Encode categorical target variable (Ex: 'smoker' column)
         label encoder = LabelEncoder()
         data['smoker'] = label encoder.fit transform(data['smoker']) # Convert data into binary form (yes/no -> 1/0)
         data['sex'] = label encoder.fit transform(data['sex'])
In [40]: # Select two numerical features for visualization (Ex. Age & BMI)
         selected_features = ["age","bmi"]
         X = data[selected features].values # Convert to numpy array
         y = data["smoker"].values # Target variable (smoker classification)
In [42]: # Split into training and testing data
         X train, X test, y train, y test = train test split(X, y, test size=0.2, random state=42)
In [44]: # Apply Standar Scaling on selected features
         scaler = StandardScaler()
         X train = scaler.fit transform(X train) # Fit and transform training data
         X test = scaler.transform(X test) # Fit and transform testing data
In [46]: print(f"Training data shape: {X train.shape}")
         print(f"Testing data shape: {X test.shape}")
        Training data shape: (1070, 2)
        Testing data shape: (268, 2)
In [48]: knn = KNeighborsClassifier(n neighbors=3)
         knn.fit(X_train, y_train)
Out[48]: 🔻
                KNeighborsClassifier
         KNeighborsClassifier(n neighbors=3)
In [50]: # Create a mesh grid for decision boundary
         x_{min}, x_{max} = X[:, 0].min() - 1, <math>X[:, 0].max() + 1
         y_{min}, y_{max} = X[:, 1].min() - 1, X[:, 1].max() + 1
```

```
xx, yy = np.meshgrid(np.arange(x_min, x_max, 200), np.linspace(y_min, y_max, 200))

In [52]: # Ensure mesh points are transformed using the same scaler
    mesh_points = np.c_[xx.ravel(),yy.ravel()]
    Z = knn.predict(scaler.transform(mesh_points))

In [56]: sns.scatterplot(x=X[:,0], y=X[:,1], hue=y, palette='coolwarm',edgecolor='black')
    plt.xlabel("Age")
    plt.ylabel("BMI")
    plt.title("KNN Classification with Decision Boundaries for Insurance Data")
    plt.legend(["Non Smoker (0)", "Smoker (1)"])
    plt.show()
```

## KNN Classification with Decision Boundaries for Insurance Data



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js