Dataset: We have a simple dataset with mixed features:

Height (numerical) Gender (categorical) Weight (numerical) Age (numerical)

In [1]:

```
import numpy as np
from collections import Counter
from sklearn.preprocessing import LabelEncoder

class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def euclidean_distance(self, x1, x2):
        return np.sqrt(np.sum((x1 - x2)**2))

    def predict_single(self, query):
        # Compute distances between query and all examples in the training
        distances = [self.euclidean_distance(query, x_train) for x_train in

        # Sort by distance and return indices of the first k neighbors
        k_indices = np.argsort(distances)[:self.k]

        # Extract the labels of the k nearest neighbor training samples
        k_nearest_labels = [self.y_train[i] for i in k_indices]

        # Perform majority vote, most common class label among the k neighb
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

# Example usage with a dataset
if __name__ == "__main__":
    # Sample dataset with mixed numerical and categorical features
    X_train = np.array([[6, 'M', 180, 12], [5.5, 'F', 150, 8], [6.1, 'M', 1
                        [5.9, 'F', 160, 10], [6.2, 'M', 175, 14], [5.8, 'F'
    y_train = np.array(['Tall', 'Short', 'Tall', 'Short', 'Tall', 'Short'])

    X_test = np.array([[6, 'F', 155, 11], [5.7, 'M', 170, 13]])

    # Initialize the KNN classifier
    knn = KNN(k=3)

    # Encode categorical features in both X_train and X_test
    label_encoders = []
    for i in range(X_train.shape[1]):
        if isinstance(X_train[0, i], str):
            le = LabelEncoder()
            combined_data = np.concatenate((X_train[:, i], X_test[:, i]), a
            le.fit(combined_data)
            X_train[:, i] = le.transform(X_train[:, i])
            X_test[:, i] = le.transform(X_test[:, i])
            label_encoders.append(le)

    # Fit the KNN model with the encoded training data
    knn.fit(X_train.astype(float), y_train)

    # Define a specific query
    query = np.array([6, 'F', 155, 11])

    # Encode the query using the same label encoders
    for i, le in enumerate(label_encoders):
        if isinstance(query[i], str):
            query[i] = le.transform([query[i]])[0]

    # Predict the label of the query
    predicted_class = knn.predict_single(query.astype(float))

    # Print result
    print(f"Query predicted class: {predicted_class}")
```

```
Query predicted class: Tall
```

In [ ]: