```
# Import the required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
# Load the dataset
data = pd.read_csv('/content/health_data - Sheet1.csv')  # Replace 'data.csv' with the actual filename of your dataset
```

```
# Split the dataset into features (X) and labels (y)
X = data.iloc[:, :-1]  # Select all columns except the last one as features
y = data.iloc[:, -1]  # Create labels based on the values in the last column
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create an SVM classifier
from sklearn import svm
clf = svm.SVC()  # You can choose different kernels like 'rbf' or 'poly' depending on your dataset
```

```
# Train the classifier
clf.fit(X_train, y_train)
```

```
     ▾ SVC
     SVC()
```

```
# Make predictions on the test set
y_pred = clf.predict(X_test)
```

```
y_pred
```

```
     array(['UnHealthy', 'Healthy', 'Healthy', 'Healthy', 'Healthy', 'Healthy',
            'Healthy', 'Healthy', 'Healthy', 'Healthy', 'Healthy', 'Healthy',
            'Healthy', 'Healthy', 'UnHealthy', 'Healthy'], dtype=object)
```

```
# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
     Accuracy: 1.0
```

```
import numpy as np

new_data = np.array([[20,60,37]])
```

```
# Get the decision function scores
decision_scores = clf.decision_function(new_data)
```

```
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was fitted with f
       warnings.warn(
```

```
# Apply lower and upper bounds to the decision scores
lower_bounds = [90,55,30]  # Lower bounds for each feature column
upper_bounds = [101,101,38]     # Upper bounds for each feature column
```

```
clipped_decision_scores = np.clip(decision_scores, lower_bounds, upper_bounds)
```

```
# Check if each feature value is within the specified bounds
validity = [lower <= value <= upper for lower, upper, value in zip(lower_bounds, upper_bounds, new_data[0])]
```

```
# If all features are within bounds, predict as "Healthy"; otherwise, predict as "Not Healthy"
if all(validity):
    prediction = " Person is Healthy"
else:
    prediction = "Person is Not Healthy"

print("Prediction:", prediction)
```

```
Prediction: Person is Not Healthy
```

```
# Create an array of predictions for each feature's validity status specific parameter
predictions = ["Healthy" if valid else "Not Healthy" for valid in validity]

print("Predictions for each feature:", predictions)
```

```
Predictions for each feature: ['Not Healthy', 'Healthy', 'Healthy']
```