```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import joblib

ranges = {
    "pH": (6.5, 8.5),
    "Hardness": (0, 300),
    "Solids": (0, 500),
    "Chloramines": (0, 4),
    "Sulfate": (0, 250),
    "Conductivity": (0, 400),
    "Organic Carbon": (0, 5),
    "Trihalomethanes": (0, 80),
    "Turbidity": (0, 1),
}

def prepare_and_train_model():
    file_path = '/content/water_potability.csv'
    data = pd.read_csv(file_path)

    imputer = SimpleImputer(strategy='mean')
    data_imputed = imputer.fit_transform(data)

    X = data_imputed[:, :-1]
    y = data_imputed[:, -1]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model = SVC()
    model.fit(X_train, y_train)

    joblib.dump(model, 'water_quality_svm_model.pkl')

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model accuracy: {accuracy:.2f}")

def load_model():
    return joblib.load('water_quality_svm_model.pkl')

def predict_water_quality(model):
    try:
        ph = float(input(f"Enter pH ({ranges['pH'][0]} - {ranges['pH'][1]}): "))
        hardness = float(input(f"Enter Hardness ({ranges['Hardness'][0]} - {ranges['Hardness'][1]} mg/L): "))
        solids = float(input(f"Enter Solids ({ranges['Solids'][0]} - {ranges['Solids'][1]} mg/L): "))
        chloramines = float(input(f"Enter Chloramines ({ranges['Chloramines'][0]} - {ranges['Chloramines'][1]} mg/L): "))
        sulfate = float(input(f"Enter Sulfate ({ranges['Sulfate'][0]} - {ranges['Sulfate'][1]} mg/L): "))
        conductivity = float(input(f"Enter Conductivity ({ranges['Conductivity'][0]} - {ranges['Conductivity'][1]} µS/cm): "))
        organic_carbon = float(input(f"Enter Organic Carbon ({ranges['Organic Carbon'][0]} - {ranges['Organic Carbon'][1]} mg/L): ")
        trihalomethanes = float(input(f"Enter Trihalomethanes ({ranges['Trihalomethanes'][0]} - {ranges['Trihalomethanes'][1]} µg/L)
        turbidity = float(input(f"Enter Turbidity ({ranges['Turbidity'][0]} - {ranges['Turbidity'][1]} NTU): "))

        input_data = [[ph, hardness, solids, chloramines, sulfate, conductivity, organic_carbon, trihalomethanes, turbidity]]
        prediction = model.predict(input_data)
        result = "Potable" if prediction[0] == 1 else "Not Potable"
        print(f"The water is {result}")

        within_range = all([
            ranges['pH'][0] <= ph <= ranges['pH'][1],
            ranges['Hardness'][0] <= hardness <= ranges['Hardness'][1],
            ranges['Solids'][0] <= solids <= ranges['Solids'][1],
            ranges['Chloramines'][0] <= chloramines <= ranges['Chloramines'][1],
            ranges['Sulfate'][0] <= sulfate <= ranges['Sulfate'][1],
            ranges['Conductivity'][0] <= conductivity <= ranges['Conductivity'][1],
            ranges['Organic Carbon'][0] <= organic_carbon <= ranges['Organic Carbon'][1],
            ranges['Trihalomethanes'][0] <= trihalomethanes <= ranges['Trihalomethanes'][1],
            ranges['Turbidity'][0] <= turbidity <= ranges['Turbidity'][1]
        ])

        if within_range:
            print("The entered values are within the range for good water quality.")
        else:
            print("One or more values are outside the range for good water quality.")

    except ValueError:
        print("Invalid input. Please enter numeric values for all parameters.")

prepare_and_train_model()
```

```
model = load_model()

predict_water_quality(model)
```

```
Model accuracy: 0.63
Enter pH (6.5 - 8.5): 7
Enter Hardness (0 - 300 mg/L): 350
Enter Solids (0 - 500 mg/L): 400
Enter Chloramines (0 - 4 mg/L): 3
Enter Sulfate (0 - 250 mg/L): 200
Enter Conductivity (0 - 400 µS/cm): 300
Enter Organic Carbon (0 - 5 mg/L): 3
Enter Trihalomethanes (0 - 80 µg/L): 24
Enter Turbidity (0 - 1 NTU): 1
The water is Not Potable
One or more values are outside the range for good water quality.
```