

**Availability of medicines and doctors in government hospitals.**

A predictive model is trained for both doctor availability and medicine stock. The system provides predictions alongside the static checks. We will use Random forest models for demonstration purposes

```
import numpy as np
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.preprocessing import LabelEncoder

# Load the dataset
file_path = '/content/Book 4.xlsx'
xls = pd.ExcelFile(file_path)
data = xls.parse(xls.sheet_names[0])

# Set up dictionaries for doctors and medicines
doctors = {
    row['Doctors']: {
        "available": random.choice([True, False]), # Randomly assigning availability status
        "hospital": row['Hospitals'],
        "specialist": row['Specialist']
    }
    for _, row in data.iterrows()
}

medicines = {
    row['Medicines']: {
        "available": row['Stock Quantity'] > 0, # Available if stock is greater than 0
        "stock": row['Stock Quantity'],
        "Price": row['Price']
    }
    for _, row in data.iterrows()
}

# Preprocessing for ML Models
data['DoctorAvailable'] = data['Doctors'].apply(lambda x: random.choice([1, 0])) # Random binary labels for demonstration
data['MedicineAvailable'] = data['Stock Quantity'] > 0 # Binary label for medicine availability

# Encode categorical features
label_encoders = {}
categorical_columns = ['Doctors', 'Hospitals', 'Specialist', 'Medicines']
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Features and targets for doctors and medicines
doctor_features = data[['Doctors', 'Hospitals', 'Specialist']]
doctor_target = data['DoctorAvailable']

medicine_features = data[['Medicines', 'Stock Quantity', 'Price']]
medicine_target = data['MedicineAvailable']

# Train-test split
X_train_doctors, X_test_doctors, y_train_doctors, y_test_doctors = train_test_split(
    doctor_features, doctor_target, test_size=0.2, random_state=42)
X_train_meds, X_test_meds, y_train_meds, y_test_meds = train_test_split(
    medicine_features, medicine_target, test_size=0.2, random_state=42)

# Train models
doctor_model = RandomForestClassifier(random_state=42)
doctor_model.fit(X_train_doctors, y_train_doctors)

medicine_model = RandomForestRegressor(random_state=42)
medicine_model.fit(X_train_meds, y_train_meds)
```



```
# Function to check availability of doctors with predictions
def check_doctor_availability():
    print("\nAvailable Doctors:")
    for doctor, details in doctors.items():
        # Prepare input for prediction
        doctor_row = pd.DataFrame([
            "Doctors": label_encoders['Doctors'].transform([doctor])[0],
            "Hospitals": label_encoders['Hospitals'].transform([details['hospital']])[0],
            "Specialist": label_encoders['Specialist'].transform([details['specialist']])[0]
        ])
        prediction = doctor_model.predict(doctor_row)[0]
        status = "Available" if prediction == 1 else "Unavailable"
        print(f"{doctor} (Hospital: {details['hospital']}, Specialist: {details['specialist']}) - {status}")

# Function to check availability of medicines with predictions
def check_medicine_availability():
    print("\nAvailable Medicines:")
    for medicine, details in medicines.items():
        # Prepare input for prediction
        med_row = pd.DataFrame([
            "Medicines": label_encoders['Medicines'].transform([medicine])[0],
            "Stock Quantity": details['stock'],
            "Price": details['Price'] # Changed 'price' to 'Price' to match the training data
        ])
        prediction = medicine_model.predict(med_row)[0]
        status = "Available" if prediction > 0.5 else "Unavailable"
        print(f"{medicine}: {status} (Stock: {details['stock']}, Price: {details['Price']})")

# Main menu function
def main_menu():
    while True:
        print("\n--- Healthcare Availability Menu ---")
        print("1. Check Doctor Availability")
        print("2. Check Medicine Availability")
        print("3. Exit")

        choice = input("Enter your choice (1-3): ")

        if choice == "1":
            check_doctor_availability()
        elif choice == "2":
            check_medicine_availability()
        elif choice == "3":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

# Run the main menu
main_menu()
```



```

Naproxen: Available (Stock: 50, Price: 3222),
Naproxen: Available (Stock: 40, Price: 3834)
Mefenamic Acid: Available (Stock: 30, Price: 2133)
Dexamethasone: Available (Stock: 60, Price: 3333)
Prednisone: Available (Stock: 20, Price: 2223)
Paracetamol + Caffeine: Available (Stock: 85, Price: 2125)
Caffeine: Available (Stock: 45, Price: 6671)
Vitamin C: Available (Stock: 50, Price: 3521)
Zinc: Available (Stock: 25, Price: 2524)
Electrolyte Solutions: Available (Stock: 35, Price: 2275)
Oral Rehydration Salts: Available (Stock: 30, Price: 2717)
Phenobarbital: Available (Stock: 50, Price: 6285)
Loratadine + Pseudoephedrine: Available (Stock: 40, Price: 6037)
Saline nasal spray: Available (Stock: 75, Price: 4527)
Calamine lotion: Available (Stock: 30, Price: 2575)
Topical Antiseptics: Available (Stock: 50, Price: 3725)
Antihistamine eye drops: Available (Stock: 20, Price: 6026)
Loperamide: Available (Stock: 15, Price: 2678)
Probiotics: Available (Stock: 25, Price: 6225)
Antacids: Available (Stock: 40, Price: 2678)
Humidifiers: Available (Stock: 35, Price: 2435)
Lozenges: Available (Stock: 30, Price: 3789)
Cough syrups: Available (Stock: 45, Price: 4938)

```

```

--- Healthcare Availability Menu ---

```

```

1. Check Doctor Availability
2. Check Medicine Availability
3. Exit
Enter your choice (1-3): 3
Exiting...

```

```

from google.colab import drive
drive.mount('/content/drive')

```

```

# Import libraries
import pandas as pd
import random
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, mean_absolute_error, r2_score

```

```

# Load the dataset

```

```

file_path = '/content/Book 4.xlsx'
xls = pd.ExcelFile(file_path)
data = xls.parse(xls.sheet_names[0])

```

```

# Set up dictionaries for doctors and medicines

```

```

doctors = {
    row['Doctors']: {
        "available": random.choice([True, False]), # Randomly assigning availability status
        "hospital": row['Hospitals'],
        "specialist": row['Specialist']
    }
    for _, row in data.iterrows()
}

```

```

medicines = {
    row['Medicines']: {
        "available": row['Stock Quantity'] > 0, # Available if stock is greater than 0
        "stock": row['Stock Quantity'],
        "value": row['Price']
    }
    for _, row in data.iterrows()
}

```

```

# Preprocessing for ML Models

```

```

data['DoctorAvailable'] = data['Doctors'].apply(lambda x: random.choice([1, 0])) # Random binary labels for demonstration
data['MedicineAvailable'] = data['Stock Quantity'] > 0 # Binary label for medicine availability

```

```

# Encode categorical features

```

```

label_encoders = {}
categorical_columns = ['Doctors', 'Hospitals', 'Specialist', 'Medicines']
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

```

```

# Features and targets for doctors and medicines

```

```

doctor_features = data[['Doctors', 'Hospitals', 'Specialist']]
doctor_target = data['DoctorAvailable']

```

```

medicine_features = data[['Medicines', 'Stock Quantity', 'Price']]

```

```

medicine_target = data['MedicineAvailable']

# Train-test split
X_train_doctors, X_test_doctors, y_train_doctors, y_test_doctors = train_test_split(
    doctor_features, doctor_target, test_size=0.2, random_state=42)
X_train_meds, X_test_meds, y_train_meds, y_test_meds = train_test_split(
    medicine_features, medicine_target, test_size=0.2, random_state=42)

# Train models
doctor_model = RandomForestClassifier(random_state=42)
doctor_model.fit(X_train_doctors, y_train_doctors)

medicine_model = RandomForestRegressor(random_state=42)
medicine_model.fit(X_train_meds, y_train_meds)

# Evaluate Doctor Model (Classification)
y_pred_doctors = doctor_model.predict(X_test_doctors)
doctor_accuracy = accuracy_score(y_test_doctors, y_pred_doctors)
print("\n--- Doctor Model Evaluation ---")
print(f"Accuracy: {doctor_accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test_doctors, y_pred_doctors))


# Evaluate Medicine Model (Regression)
y_pred_meds = medicine_model.predict(X_test_meds)
mae = mean_absolute_error(y_test_meds, y_pred_meds)
r2 = r2_score(y_test_meds, y_pred_meds)
print("\n--- Medicine Model Evaluation ---")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R-squared (R2 Score): {r2:.2f}")

# Hyperparameter Tuning (Optional)
def tune_model(model, params, X_train, y_train):
    grid_search = GridSearchCV(model, param_grid=params, cv=5, scoring='accuracy' if isinstance(model, RandomForestClassifier) else 'neg')
    grid_search.fit(X_train, y_train)
    return grid_search.best_estimator_

# Example: Tuning RandomForestClassifier
rf_params = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10]}
tuned_doctor_model = tune_model(RandomForestClassifier(random_state=42), rf_params, X_train_doctors, y_train_doctors)

print("\n--- Tuned Doctor Model ---")
print(tuned_doctor_model)

```



```

--- Doctor Model Evaluation ---
Accuracy: 0.60
Classification Report:

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.50      | 0.50   | 0.50     | 4       |
| 1            | 0.67      | 0.67   | 0.67     | 6       |
| accuracy     |           |        | 0.60     | 10      |
| macro avg    | 0.58      | 0.58   | 0.58     | 10      |
| weighted avg | 0.60      | 0.60   | 0.60     | 10      |

```

--- Medicine Model Evaluation ---
Mean Absolute Error (MAE): 0.00
R-squared (R2 Score): 1.00

--- Tuned Doctor Model ---
RandomForestClassifier(n_estimators=50, random_state=42)

```

```

import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score

# Load the dataset
file_path = '/content/Book 4.xlsx'
xls = pd.ExcelFile(file_path)
data = xls.parse(xls.sheet_names[0])

# Generate dictionaries for doctors and medicines
doctors = {
    row['Doctors']: {
        "available": random.choice([True, False]), # Randomly assign availability
        "hospital": row['Hospitals'],
        "specialist": row['Specialist']
    }
}

```

```

    }
    for _, row in data.iterrows()
}

medicines = {
    row['Medicines']: {
        "available": row['Stock Quantity'] > 0, # Available if stock > 0
        "stock": row['Stock Quantity'],
        "Price": row['Price']
    }
    for _, row in data.iterrows()
}

# Preprocessing for Machine Learning
data['DoctorAvailable'] = data['Doctors'].apply(lambda x: random.choice([1, 0])) # Random binary labels
data['MedicineAvailable'] = data['Stock Quantity'] > 0 # Binary label for medicines

# Encode categorical features
label_encoders = {}
categorical_columns = ['Doctors', 'Hospitals', 'Specialist', 'Medicines']
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Features and targets
doctor_features = data[['Doctors', 'Hospitals', 'Specialist']]
doctor_target = data['DoctorAvailable']

medicine_features = data[['Medicines', 'Stock Quantity', 'Price']]
medicine_target = data['MedicineAvailable']

# Train-test split
X_train_doctors, X_test_doctors, y_train_doctors, y_test_doctors = train_test_split(
    doctor_features, doctor_target, test_size=0.2, random_state=42
)
X_train_meds, X_test_meds, y_train_meds, y_test_meds = train_test_split(
    medicine_features, medicine_target, test_size=0.2, random_state=42
)

# Train models
doctor_model = RandomForestClassifier(random_state=42)
doctor_model.fit(X_train_doctors, y_train_doctors)

medicine_model = RandomForestRegressor(random_state=42)
medicine_model.fit(X_train_meds, y_train_meds)

# Linear Regression Model for Medicine Price Prediction
price_data = data[['Medicines', 'Stock Quantity', 'Price']].drop_duplicates()
X = price_data[['Stock Quantity']]
y = price_data['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
price_model = LinearRegression()
price_model.fit(X_train, y_train)

# Evaluate Price Model
y_pred = price_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Functions to check availability and predict price
def check_doctor_availability():
    print("\n\nAvailable Doctors:")
    for doctor, details in doctors.items():
        doctor_row = pd.DataFrame([
            "Doctors": label_encoders['Doctors'].transform([doctor])[0],
            "Hospitals": label_encoders['Hospitals'].transform([details['hospital']])[0],
            "Specialist": label_encoders['Specialist'].transform([details['specialist']])[0]
        ])
        prediction = doctor_model.predict(doctor_row)[0]
        status = "Available" if prediction == 1 else "Unavailable"
        print(f"{doctor} (Hospital: {details['hospital']}, Specialist: {details['specialist']}) - {status}")

def check_medicine_availability():
    print("\n\nAvailable Medicines:")
    for medicine, details in medicines.items():
        med_row = pd.DataFrame([
            "Medicines": label_encoders['Medicines'].transform([medicine])[0],
            "Stock Quantity": details['stock'],
            "Price": details['Price']
        ])
        prediction = medicine_model.predict(med_row)[0]
        status = "Predicted Available" if prediction > 0.5 else "Unavailable"

```

```

status = 'Predicted Available' if prediction > 0.5 else 'Unavailable'
print(f"{medicine}: {status} (Stock: {details['stock']}, Price: {details['Price']})")

def predict_medicine_price(stock_quantity):
    new_data = pd.DataFrame({"Stock Quantity": [stock_quantity]})
    predicted_price = price_model.predict(new_data)[0]
    print(f"\nPredicted Price for stock quantity {stock_quantity}: ${predicted_price:.2f}")

# Main menu
def main_menu():
    while True:
        print("\n--- Healthcare Availability Menu ---")
        print("1. Check Doctor Availability")
        print("2. Check Medicine Availability")
        print("3. Predict Medicine Price")
        print("4. Exit")
        choice = input("Enter your choice (1-4): ")
        if choice == "1":
            check_doctor_availability()
        elif choice == "2":
            check_medicine_availability()
        elif choice == "3":
            stock_quantity = int(input("Enter stock quantity: "))
            predict_medicine_price(stock_quantity)
        elif choice == "4":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

# Run the main menu
main_menu()

```

--- Healthcare Availability Menu ---  
 1. Check Doctor Availability  
 2. Check Medicine Availability  
 3. Predict Medicine Price  
 4. Exit  
 Enter your choice (1-4): 1

Available Doctors:  
 Dr. Abhideep Chaudhary (Hospital: K.C. General Hospital, Specialist: General Physician) - Available  
 Dr. Ajaya Nand Jha (Hospital: Government District Super-Speciality Hospital, Ramanagara, Specialist: Endocrinologist) - Unavailab  
 Dr. Suresh Joshi (Hospital: Victoria Hospital, Specialist: Orthopedic Surgeon) - Available  
 Dr. P S Ragavan (Hospital: Govt. H.S.I.S Gosha Hospital, Specialist: Dermatologist) - Available  
 Dr. Surendra V H H, (Hospital: K K Hospital, Specialist: Pediatrician) - Unavailable  
 Dr. Veerendra Sandur (Hospital: PHC Govt. Hospital, Nagara Arogya Kendra, Specialist: Gynecologist) - Available  
 Dr. Padmavathi K Iyer (Hospital: Sir C V Raman General Hospital, Specialist: Neurologist) - Unavailable  
 Dr. Chandana C (Hospital: Government hospital Bagalur, Specialist: ENT Specialist) - Available  
 Dr. Manohar J (Hospital: Motherhood Hospital, Specialist: Oncologist) - Unavailable  
 Dr. Raghurama N. K. (Hospital: Govt. H.S.I.S Gosha Hospital, Specialist: Endocrinologist) - Available  
 Dr. Veerendra Sandur (Hospital: Sapthagiri Super Speciality Hospital, Specialist: General Physician) - Available  
 Dr. Padmavathi K Iyer (Hospital: Dr. B.R. Ambedkar Hospital, Specialist: Cardiologist) - Available  
 Dr. Manan Vora (Hospital: Bangalore University Health Center, Specialist: Orthopedic Surgeon) - Available  
 Dr. Sunil Richardson (Hospital: Bangalore Baptist Hospital, Specialist: Dermatologist) - Available  
 Dr. Mohit Bhandari (Hospital: MS Ramaiah Old Hospital, Specialist: Pediatrician) - Unavailable  
 Dr. Chirag Jain (Hospital: Indira Gandhi Institute Of Child Health, Specialist: Gynecologist) - Available  
 Dr. Jagdish Chaturvedi (Hospital: Trinity Central Hospital, Specialist: Neurologist) - Available  
 Dr. Dinesh Singh (Hospital: Govt Nizamia General Hospital, Specialist: ENT Specialist) - Available  
 Dr. Arunkumar M. J (Hospital: Arignar Anna Govt Hospital Of India Medicine, Specialist: Oncologist) - Available  
 Dr. Suresh Joshi (Hospital: Tamil Nadu Government Multi Super Speciality Hospital TNGMSSH, Specialist: General Physician) - Unav  
 Dr. Abhijit Dey (Hospital: ESI Hospital - Main Branch Rajajinagar, Specialist: Cardiologist) - Unavailable  
 Dr. Sandeep Vaishya (Hospital: Charminar Govt Hospital, Specialist: Orthopedic Surgeon) - Available  
 Dr. Rahul Bhargava (Hospital: Government District Hospital Nilambur, Specialist: Dermatologist) - Available  
 Dr. Deepak Namjoshi (Hospital: Government Area Hospital, Vanasthalipuram, Specialist: Pediatrician) - Available  
 Dr. Sheetal Kamat (Hospital: Government Maternity Hospital (GMH SB), Sultan Bazar, Specialist: Gynecologist) - Available  
 Dr. Sujit Kumar (Hospital: Cytecure Hospitals Bangalore, Specialist: Neurologist) - Available  
 Dr. Shankar V (Hospital: Govt General and Chest Hospital, Specialist: ENT Specialist) - Available  
 Dr. Jaya Ranganath (Hospital: Government ENT Hospital, Specialist: Cardiologist) - Unavailable  
 Dr. Pradeep N (Hospital: East Point Hospital, Specialist: Endocrinologist) - Unavailable  
 Dr. Wasim Dar (Hospital: Govt general hospital TVVP(CHC) 50 Bedded, Specialist: General Physician) - Available  
 Dr. Gandikota Jameel Ahammed (Hospital: Karnataka Cancer Hospital, Specialist: Orthopedic Surgeon) - Available  
 Dr. Manjula J (Hospital: Government Fever Hospital, Specialist: Dermatologist) - Unavailable  
 Dr. Manish Joshi (Hospital: Ramaiah Narayana Heart Centre, M S Ramaiah Nagar, Specialist: Pediatrician) - Available  
 Dr. Aishwarya Lakshmi B. R (Hospital: Banglore Holistic Medical Center, Specialist: Gynecologist) - Available  
 Dr. Zankhana M Buch (Hospital: Dr Agarwals Eye Hospital, Specialist: Neurologist) - Unavailable  
 Dr. Savitha Shetty (Hospital: Narayana Super Specialty Hospitals, Specialist: ENT Specialist) - Unavailable  
 Dr. Pradeep Hosamani H (Hospital: BMS Hospital, Specialist: Oncologist) - Available  
 Dr. Palaniappan Ramanathan (Hospital: Government Taluk Hospital Kayamkulam, Specialist: Endocrinologist) - Unavailable  
 Dr. Murali Mohan C R (Hospital: Agarahara Government Hospital, Specialist: General Physician) - Available  
 Dr. Sunil S Bohra (Hospital: Thanisandra Government Hospital, Specialist: Cardiologist) - Available  
 Dr. C A Prashanth (Hospital: Jayanagar General Hospital, Specialist: Orthopedic Surgeon) - Unavailable  
 Dr. C A Prashanth (Hospital: PHC Govt. Hospital, Nagara Arogya Kendra, Specialist: Dermatologist) - Available  
 Dr. Sahana K P (Hospital: Bangalore Mahanagar Palika Government Hospital, Specialist: Pediatrician) - Available

Dr. Prasanna Katti (Hospital: The North Bangalore Hospital, Specialist: Gynecologist) - Available  
 Dr. N. Aditya Murali (Hospital: ESIC model Hospital Peenya, Specialist: Neurologist) - Unavailable  
 Dr. Divya K S (Hospital: Urban Primary Health Care Centre - Ganganagar, Specialist: ENT Specialist) - Available

```
doctors = {
    row['Doctors']: {
        "available": random.choice([True, False]), # Randomly assign availability
        "hospital": row['Hospitals'],
        "specialist": row['Specialist']
    }
    for _, row in data.iterrows()
}

medicines = {
    row['Medicines']: {
        "available": row['Stock Quantity'] > 0, # Available if stock > 0
        "stock": row['Stock Quantity'],
        "Price": row['Price']
    }
    for _, row in data.iterrows()
}

# Preprocessing for Machine Learning
data['DoctorAvailable'] = data['Doctors'].apply(lambda x: random.choice([1, 0])) # Random binary labels
data['MedicineAvailable'] = data['Stock Quantity'] > 0 # Binary label for medicines

# Encode categorical features
label_encoders = {}
categorical_columns = ['Doctors', 'Hospitals', 'Specialist', 'Medicines']
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Features and targets
doctor_features = data[['Doctors', 'Hospitals', 'Specialist']]
doctor_target = data['DoctorAvailable']

medicine_features = data[['Medicines', 'Stock Quantity', 'Price']]
medicine_target = data['MedicineAvailable']

# Train-test split
X_train_doctors, X_test_doctors, y_train_doctors, y_test_doctors = train_test_split(
    doctor_features, doctor_target, test_size=0.2, random_state=42
)
X_train_meds, X_test_meds, y_train_meds, y_test_meds = train_test_split(
    medicine_features, medicine_target, test_size=0.2, random_state=42
)

# Train models
doctor_model = RandomForestClassifier(random_state=42)
doctor_model.fit(X_train_doctors, y_train_doctors)

medicine_model = RandomForestRegressor(random_state=42)
medicine_model.fit(X_train_meds, y_train_meds)

# Linear Regression Model for Medicine Price Prediction
price_data = data[['Medicines', 'Stock Quantity', 'Price']].drop_duplicates()
X = price_data[['Stock Quantity']]
y = price_data['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
price_model = LinearRegression()
price_model.fit(X_train, y_train)
```

RandomForestRegressor

RandomForestRegressor(random\_state=42)

LinearRegression

LinearRegression()

```

# Evaluate Price Model
y_pred = price_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Functions to check availability and predict price
def check_doctor_availability():
    print("\nAvailable Doctors:")
    for doctor, details in doctors.items():
        doctor_row = pd.DataFrame([
            "Doctors": label_encoders['Doctors'].transform([doctor])[0],
            "Hospitals": label_encoders['Hospitals'].transform([details['hospital']])[0],
            "Specialist": label_encoders['Specialist'].transform([details['specialist']])[0]
        ])
        prediction = doctor_model.predict(doctor_row)[0]
        status = "Available" if prediction == 1 else "Unavailable"
        print(f"{doctor} (Hospital: {details['hospital']}, Specialist: {details['specialist']}) - {status}")

def check_medicine_availability():
    print("\nAvailable Medicines:")
    for medicine, details in medicines.items():
        med_row = pd.DataFrame([
            "Medicines": label_encoders['Medicines'].transform([medicine])[0],
            "Stock Quantity": details['stock'],
            "Price": details['Price']
        ])
        prediction = medicine_model.predict(med_row)[0]
        status = "Predicted Available" if prediction > 0.5 else "Unavailable"
        print(f"{medicine}: {status} (Stock: {details['stock']}, Price: {details['Price']})")

def predict_medicine_price(stock_quantity):
    new_data = pd.DataFrame({"Stock Quantity": [stock_quantity]})
    predicted_price = price_model.predict(new_data)[0]
    print(f"\nPredicted Price for stock quantity {stock_quantity}: ${predicted_price:.2f}")

import matplotlib.pyplot as plt
import seaborn as sns

# Feature importance for the doctor model
doctor_importances = doctor_model.feature_importances_
doctor_feature_names = doctor_features.columns


plt.figure(figsize=(8, 6))
sns.barplot(x=doctor_importances, y=doctor_feature_names, palette="viridis")
plt.title("Feature Importance for Doctor Model")
plt.xlabel("Importance")
plt.ylabel("Features")
plt.show()

# Feature importance for the medicine model
medicine_importances = medicine_model.feature_importances_
medicine_feature_names = medicine_features.columns

plt.figure(figsize=(8, 6))
sns.barplot(x=medicine_importances, y=medicine_feature_names, palette="plasma")
plt.title("Feature Importance for Medicine Model")
plt.xlabel("Importance")
plt.ylabel("Features")
plt.show()

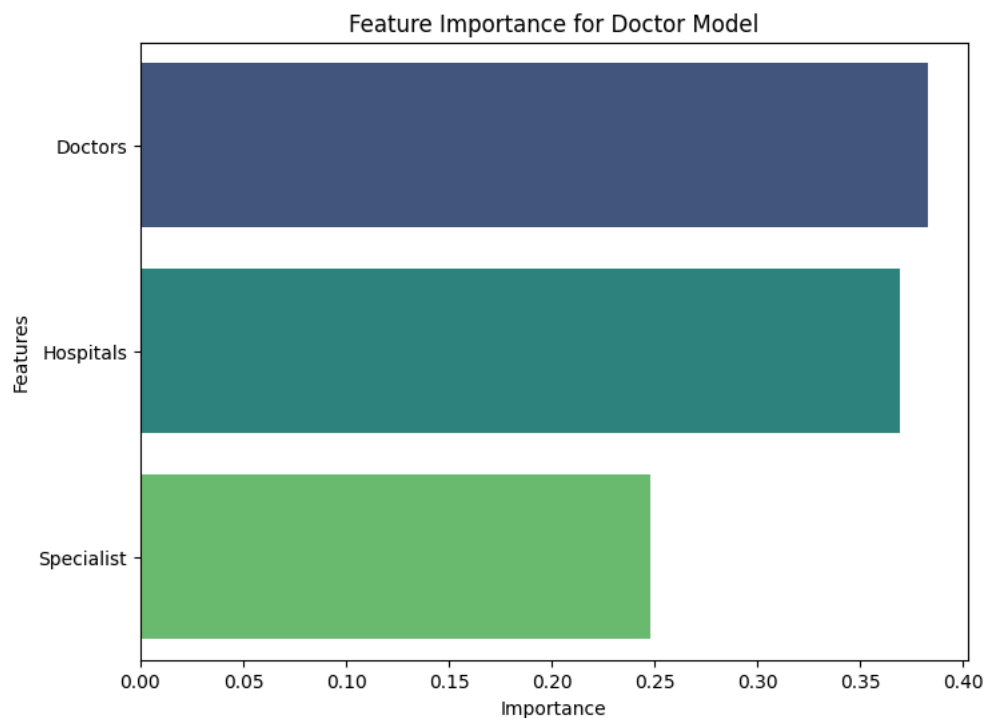
```



 <ipython-input-19-166ad6cae7b6>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

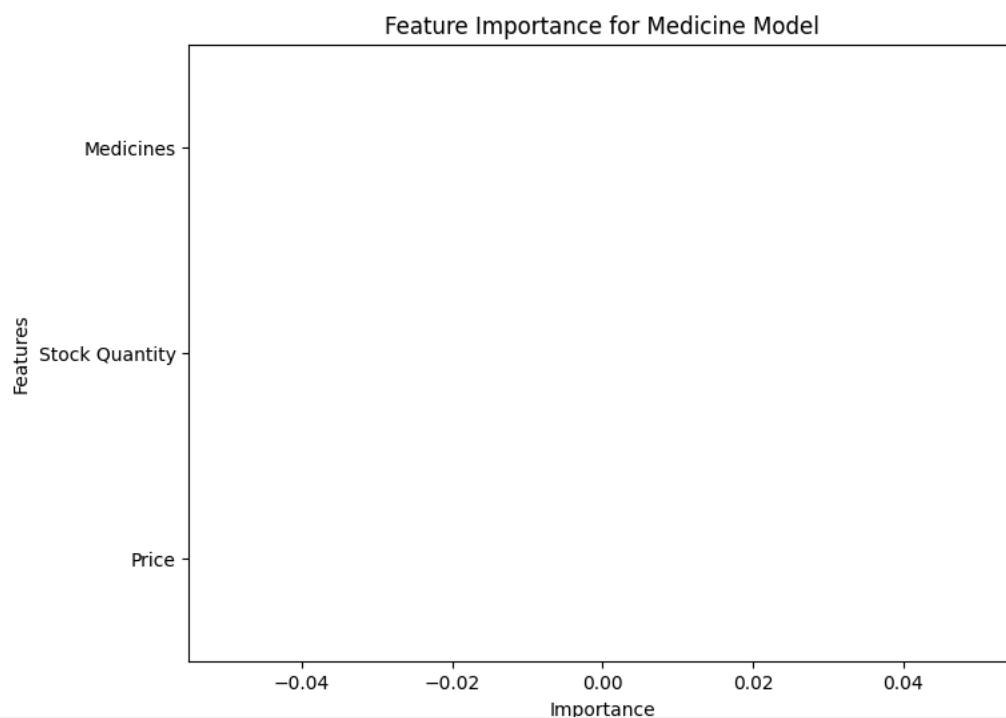
```
sns.barplot(x=doctor_importances, y=doctor_feature_names, palette="viridis")
```



<ipython-input-19-166ad6cae7b6>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

```
sns.barplot(x=medicine_importances, y=medicine_feature_names, palette="plasma")
```



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
# Predict on test data
```

```
y_pred_doctors = doctor_model.predict(X_test_doctors)
```

```
# Create confusion matrix
```

```
conf_matrix = confusion_matrix(y_test_doctors, y_pred_doctors)
```

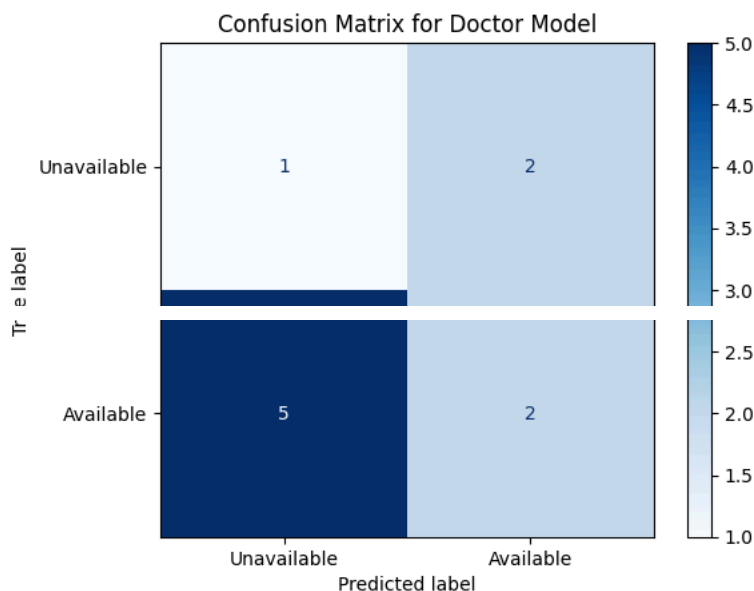
```
disp = ConfusionMatrixDisplay(conf_matrix, display_labels=["Unavailable", "Available"])
```

```
plt.figure(figsize=(6, 6))
```

```
disp.plot(cmap="Blues", values_format="d")
```

```
plt.title("Confusion Matrix for Doctor Model")
```

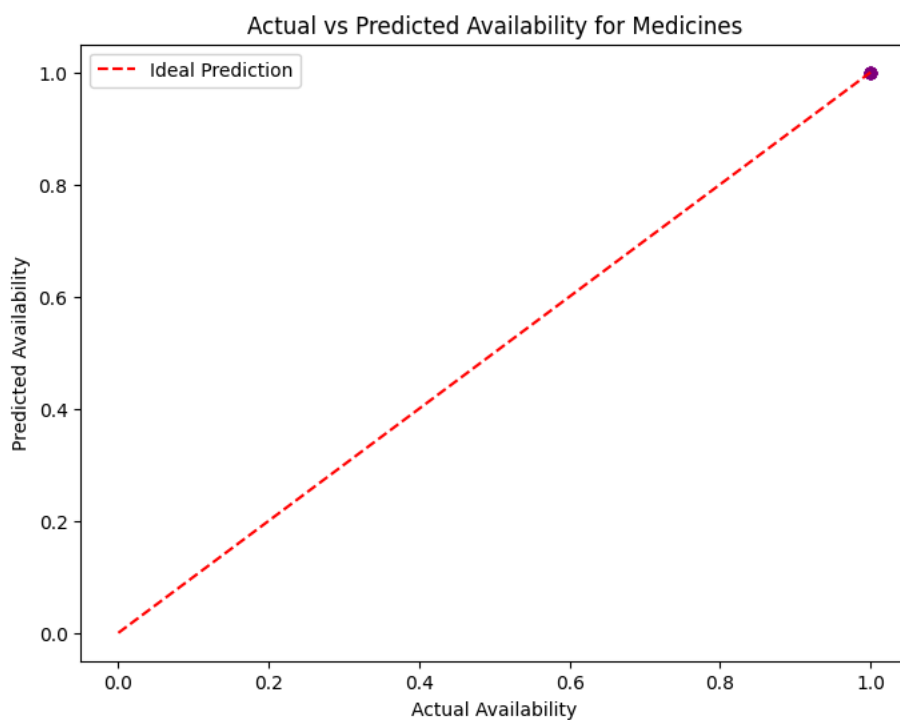
 <Figure size 600x600 with 0 Axes>



```
# Predict on test data
y_pred_meds = medicine_model.predict(X_test_meds)

plt.figure(figsize=(8, 6))
plt.scatter(y_test_meds, y_pred_meds, alpha=0.7, color='purple')
plt.plot([0, 1], [0, 1], 'r--', label="Ideal Prediction") # Line for perfect predictions
plt.title("Actual vs Predicted Availability for Medicines")
plt.xlabel("Actual Availability")
plt.ylabel("Predicted Availability")
plt.legend()
plt.show()
```





Double click (or enter) to edit