

Sistema di Predizione del Rischio Cardiovascolare

Guida Completa all'Installazione e all'Avvio

Prof. Fedeli Massimo

Versione 1.0 - 18 gennaio 2026

Basato su Regressione Logistica

Indice

1 Requisiti di Sistema	2
1.1 Software Necessario	2
1.2 Verifica Installazione Python	2
2 Installazione delle Dipendenze	2
3 Struttura del Progetto	2
4 Configurazione e Avvio del Backend	2
4.1 Passo 1: Crea il file app.py	2
4.2 Passo 2: Avvia il server	4
5 Configurazione e Avvio del Frontend	4
5.1 Passo 1: Crea il file index.html	4
5.2 Passo 2: Apri il file HTML	5
6 Test del Sistema	5
6.1 Test Rapido con Dati Precompilati	5
6.2 Parametri del Paziente di Esempio	5
7 Risoluzione Problemi Comuni	5
7.1 Problema 1: Errore di connessione al server	5
7.2 Problema 2: Modulo non trovato	5
7.3 Problema 3: Errore CORS nel browser	6
7.4 Problema 4: Porta occupata	6
8 Informazioni sul Modello	6
9 Utilizzo del Sistema	6

1 Requisiti di Sistema

1.1 Software Necessario

- **Python 3.7 o superiore** - Linguaggio di programmazione per il backend
- **pip** - Gestore pacchetti Python (solitamente incluso con Python)
- **Browser Web moderno** - Chrome, Firefox, Safari o Edge
- **Editor di testo** - VS Code, Sublime Text, o qualsiasi editor

1.2 Verifica Installazione Python

Per verificare se Python è installato correttamente, esegui nel terminale:

```
python --version  
# oppure  
python3 --version
```

Nota: Se Python non è installato, scaricalo da <https://www.python.org/downloads/>

2 Installazione delle Dipendenze

Apri il terminale o prompt dei comandi e installa i pacchetti necessari:

```
pip install flask flask-cors pandas scikit-learn
```

Pacchetti installati:

- **flask** - Framework web per il server API
- **flask-cors** - Gestione Cross-Origin Resource Sharing
- **pandas** - Manipolazione e analisi dati
- **scikit-learn** - Libreria di Machine Learning

3 Struttura del Progetto

Crea una cartella per il progetto e organizza i file come segue:

```
cardiovascular-prediction/  
|  
|-- app.py # Server Flask (Backend)  
|-- index.html # Interfaccia web (Frontend)  
|-- README.md # Documentazione (opzionale)
```

4 Configurazione e Avvio del Backend

4.1 Passo 1: Crea il file app.py

Crea un nuovo file chiamato `app.py` e copia il seguente codice:

```
"""
API Flask per Predizione Rischio Cardiovascolare
"""

from flask import Flask, request, jsonify
from flask_cors import CORS
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

app = Flask(__name__)
CORS(app)

print("Caricamento del modello...")

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"

colonne = [
    'age', 'sex', 'cp', 'trestbps', 'chol', 'fbs',
    'restecg', 'thalach', 'exang', 'oldpeak',
    'slope', 'ca', 'thal', 'target'
]

df = pd.read_csv(url, names=colonne, na_values='?')
df = df.dropna()
df['target'] = (df['target'] > 0).astype(int)

X = df.drop('target', axis=1)
y = df['target']

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

print("Modello addestrato e pronto!")

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.json

        paziente = pd.DataFrame([{
            'age': data['age'],
            'sex': data['sex'],
            'cp': data['cp'],
            'trestbps': data['trestbps'],
            'chol': data['chol'],
            'fbs': data['fbs'],
            'restecg': data['restecg'],
            'thalach': data['thalach'],
            'exang': data['exang'],
            'oldpeak': data['oldpeak'],
            'slope': data['slope'],
            'ca': data['ca'],
            'thal': data['thal'],
            'target': data['target']
        }])
    except Exception as e:
        return jsonify({'error': str(e)}), 400

    prediction = model.predict(paziente)

    if prediction[0] == 1:
        return jsonify({'rischio': 'Alto', 'probabilità': float(prediction[0])})
    else:
        return jsonify({'rischio': 'Basso', 'probabilità': float(prediction[0])})
```

```

        'restecg': data['restecg'],
        'thalach': data['thalach'],
        'exang': data['exang'],
        'oldpeak': data['oldpeak'],
        'slope': data['slope'],
        'ca': data['ca'],
        'thal': data['thal']
    }])

paziente_scaled = scaler.transform(paziente)

predizione = model.predict(paziente_scaled)[0]
probabilita = model.predict_proba(paziente_scaled)[0][1]

return jsonify({
    'prediction': int(predizione),
    'probability': float(probabilita),
    'risk': 'high' if predizione == 1 else 'low',
    'message': 'A RISCHIO' if predizione == 1 else 'SANO'
})

except Exception as e:
    return jsonify({'error': str(e)}), 400

@app.route('/health', methods=['GET'])
def health():
    return jsonify({'status': 'ok', 'model': 'Logistic Regression'})

if __name__ == '__main__':
    app.run(debug=True, port=5000)

```

4.2 Passo 2: Avvia il server

Nel terminale, dalla cartella del progetto, esegui:

```
python app.py
```

Output atteso:

```

Caricamento del modello...
Modello addestrato e pronto!
* Running on http://127.0.0.1:5000
* Restarting with stat
* Debugger is active!

```

5 Configurazione e Avvio del Frontend

5.1 Passo 1: Crea il file index.html

Il codice HTML completo è già stato fornito. Salvalo come `index.html` nella stessa cartella del progetto.

5.2 Passo 2: Apri il file HTML

- Fai doppio clic su `index.html`
- Oppure aprilo con il browser (File → Apri File)

6 Test del Sistema

6.1 Test Rapido con Dati Precompilati

1. Il form è già precompilato con i dati di un paziente di esempio
2. Clicca direttamente su “**Calcola Rischio**”
3. Dovresti vedere il risultato della predizione con la probabilità

6.2 Parametri del Paziente di Esempio

Parametro	Valore	Descrizione
Età	60	Anni
Sesso	Maschile	1 = M, 0 = F
Tipo Dolore Toracico	Angina Tipica	0-3
Pressione Riposo	150	mm Hg
Colesterolo	250	mg/dl
Glicemia Digiuno	Sì (maggiore 120)	1 = Sì, 0 = No
FC Massima	130	battiti/min
Oldpeak	2.3	Depressione ST

Tabella 1: Parametri paziente di test

7 Risoluzione Problemi Comuni

7.1 Problema 1: Errore di connessione al server

Errore: “Assicurati che il server Python sia in esecuzione”

Soluzione:

- Verifica che il server Flask sia avviato (controlla il terminale)
- Assicurati che sia in esecuzione su `http://localhost:5000`
- Se usi una porta diversa, modifica `API_URL` nell’HTML

7.2 Problema 2: Modulo non trovato

Errore: “ModuleNotFoundError: No module named ‘flask’”

Soluzione:

```
pip install flask flask-cors pandas scikit-learn
```

7.3 Problema 3: Errore CORS nel browser

Soluzione:

- Verifica che `flask-cors` sia installato
- Controlla che `CORS(app)` sia presente in `app.py`

7.4 Problema 4: Porta occupata

Soluzione:

La porta 5000 potrebbe essere occupata. Modifica la porta in `app.py`:

```
app.run(debug=True, port=5001) # Usa 5001  
invece di 5000
```

E aggiorna l'URL nell'HTML:

```
const API_URL = 'http://localhost:5001/  
predict';
```

8 Informazioni sul Modello

Algoritmo Utilizzato

Regressione Logistica - Un algoritmo di classificazione binaria che stima la probabilità che un paziente sia a rischio di malattia cardiovascolare.

Dataset

UCI Machine Learning Repository - Cleveland Heart Disease Database
297 pazienti dopo la pulizia dei dati mancanti

Preprocessing

- Standardizzazione delle feature (`media=0, std=1`)
- Rimozione valori mancanti
- Split 80/20 train/test stratificato

9 Utilizzo del Sistema

1. Inserisci i dati clinici del paziente nel form
2. Clicca su “Calcola Rischio”
3. Il sistema mostrerà:
 - Classificazione: A RISCHIO o SANO
 - Probabilità in percentuale
 - Raccomandazioni cliniche

Sistema di Predizione del Rischio Cardiovascolare

Basato su Machine Learning con Regressione Logistica

Per supporto tecnico o domande, consultare la documentazione di scikit-learn e Flask