Predicting Heart Attack Risk Using Machine Learning: A CRISP-DM Approach

Introduction

S

Satvik Atmakuri

3 min read · Sep 27, 2024

Heart disease is one of the leading causes of death worldwide. With the increase in data availability in healthcare, machine learning (ML) can significantly improve early prediction of heart disease, enabling preventative measures. In this article, we'll walk through the process of building a machine learning model to predict heart attack risk using the CRISP-DM methodology.

We'll use the Heart Disease Dataset and implement Logistic Regression and Random Forest models, followed by hyperparameter tuning to enhance model performance.

CRISP-DM Methodology

The Cross Industry Standard Process for Data Mining (CRISP-DM) is a robust methodology used for data mining and machine learning projects. It consists of six steps:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Let's dive into each step and how it was applied to this problem.

Step 1: Business Understanding

Objective: To predict whether a patient is at risk of a heart attack based on health parameters like age, cholesterol levels, blood pressure, etc. The goal is to create a model that helps doctors identify high-risk patients early, improving patient outcomes.

Step 2: Data Understanding

The Heart Disease Dataset contains 14 features like age, gender, blood pressure, and cholesterol levels. The target variable (`output`) is binary, indicating whether a patient is at risk of a heart attack.

Dataset Preview:

```
import pandas as pd
heart_data = pd.read_csv('/content/heart.csv')
heart_data.head()
```

age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1

Step 3: Data Preparation

Before building the models, we need to prepare the data. We'll:

- Split the data into training and test sets.
- Scale the features to ensure proper normalization.
- Handle any categorical data if needed.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Define features and target variable
X = heart_data.drop(columns='output')
```

```
y = heart_data['output']
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 4: Modeling

We'll build two machine learning models: Logistic Regression and Random Forest.

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
# Initialize models
log_reg = LogisticRegression()
rf_clf = RandomForestClassifier(random_state=42)
# Train the models
log_reg.fit(X_train_scaled, y_train)
rf_clf.fit(X_train_scaled, y_train)
```

Step 5: Evaluation

We evaluate the models using accuracy, precision, recall, F1-score, and ROC-AUC.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
def evaluate_model(y_true, y_pred):
    return {
        'Accuracy': accuracy_score(y_true, y_pred),
        'Precision': precision_score(y_true, y_pred),
        'Recall': recall_score(y_true, y_pred),
        'F1 Score': f1_score(y_true, y_pred),
        'ROC AUC': roc_auc_score(y_true, y_pred)
    }
# Evaluate Logistic Regression
log_reg_preds = log_reg.predict(X_test_scaled)
log_reg_eval = evaluate_model(y_test, log_reg_preds)
# Evaluate Random Forest
rf_clf_preds = rf_clf.predict(X_test_scaled)
rf_clf_eval = evaluate_model(y_test, rf_clf_preds)
log_reg_eval, rf_clf_eval
```

Visualizing the ROC Curve:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
log_reg_prob = log_reg.predict_proba(X_test_scaled)[:, 1]
rf_clf_prob = rf_clf.predict_proba(X_test_scaled)[:, 1]
fpr_log, tpr_log, _ = roc_curve(y_test, log_reg_prob)
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf_clf_prob)
# Plot ROC curve
plt.plot(fpr_log, tpr_log, label='Logistic Regression (AUC = {:.2f})'.format(auc
plt.plot(fpr_rf, tpr_rf, label='Random Forest (AUC = {:.2f})'.format(auc(fpr_rf,
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
```

Step 6: Hyperparameter Tuning

Now, let’s improve the model performance by tuning the hyperparameters using RandomizedSearchCV.

```
from sklearn.model_selection import RandomizedSearchCV
# Define hyperparameter grid for Random Forest
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30],
    'max_features': ['auto', 'sqrt']
}
# Perform Randomized Search
random_search = RandomizedSearchCV(estimator=rf_clf, param_distributions=param_g
random_search.fit(X_train_scaled, y_train)
# Get the best parameters and evaluate
best_rf = random_search.best_estimator_
best_rf_preds = best_rf.predict(X_test_scaled)
evaluate_model(y_test, best_rf_preds)
```

Step 7: Deployment

After tuning the models, we save the best one (Logistic Regression) for deployment.

```
pimport joblib
joblib.dump(log_reg, 'logistic_regression_heart_attack_model.pkl')
```

Conclusion

We've successfully walked through the steps of the CRISP-DM methodology to build a machine learning model for predicting heart attack risk. After preparing the data and training multiple models, we found that Logistic Regression and Random Forest performed well, and we further improved the Random Forest model using hyperparameter tuning.

By using these models, healthcare providers can better predict patient risk, leading to earlier interventions and improved patient outcomes.

Satvik Atmakuri

Building an Income Prediction Model with Machine Learning: A...

Machine learning projects are all about exploration, analysis, and iteration. In this...

Sep 27

Satvik Atmakuri

Building a Wine Quality Prediction Model Using the SEMMA...

In this article, we'll explore how to use the SEMMA methodology to build a machine...

Sep 27

Satvik Atmakuri

Building a Fraud Detection Model Using Random Forest and KDD...

Introduction

Sep 27

See all from Satvik Atmakuri

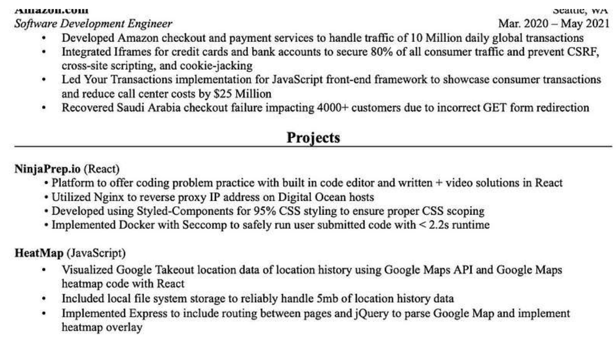
Recommended from Medium

Stephen Echessa

Stacking Ensembles: Combining XGBoost, LightGBM and CatBoost...

In the ever-evolving world of machine learning, where numerous algorithms vie for...

Jul 29 · 24 · 1



Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

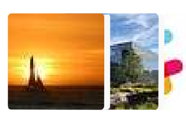
1-page. Well-formatted.

May 31 · 25K · 478

Lists

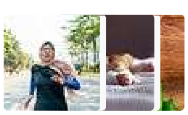
Staff Picks

755 stories · 1416 saves



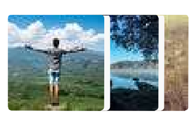
Stories to Help You Level-Up at Work

19 stories · 852 saves



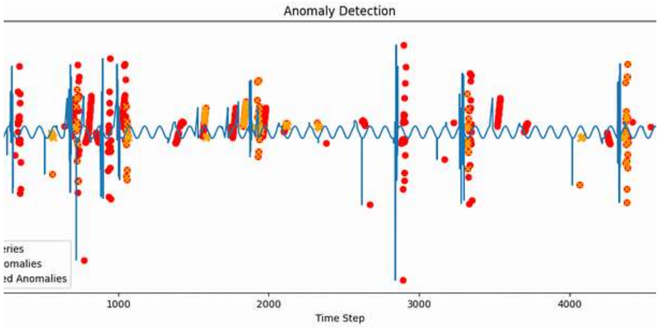
Self-Improvement 101

20 stories · 2957 saves



Productivity 101

20 stories · 2505 saves

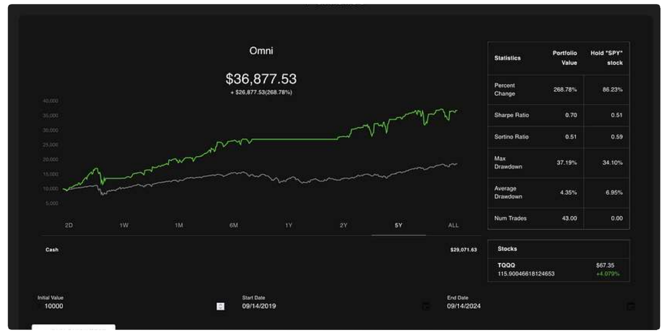


Ronan Takizawa

Time Series Anomaly Detection using Kolmogorov-Arnold...

How to build a KAN for Time Series Anomaly Detection

6d ago 61 1



Austin Starks in DataDrivenInvestor

I used OpenAI's o1 model to develop a trading strategy. It is...

It literally took one try. I was shocked.

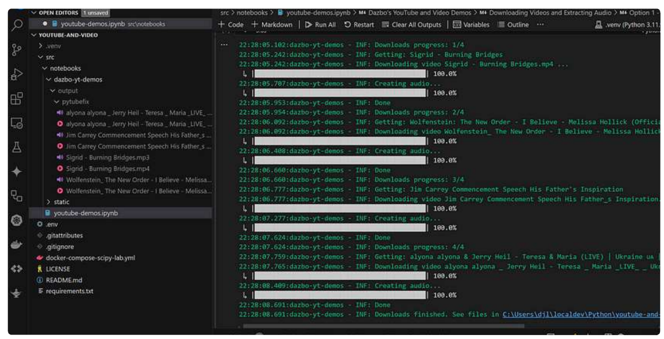
Sep 15 5.3K 138

Ayomitan Adesua in The Deep Hub

Predicting and Explaining Customer Churn: A Data Science...

How Data Science and Causal Inference Can Help Predict and Reduce Customer Churn to...

Oct 1 209 2



Dazbo (Darren Lester) in Python in Plain English

Downloading YouTube Videos, Extracting Audio, and Generating...

Demonstrating Python Jupyter notebook to: download videos from YouTube, extract audi...

Oct 24 737 5

See more recommendations