Course Name: DBMS Lab

Course Code: CSEG2146

Faculty Name: Mr. Syed Sajid Hussain

Submitted By:

Anoushka Pandey

Batch: B1

Sap Id: 500120245

Roll no.: R2142230030

Experiment 22:

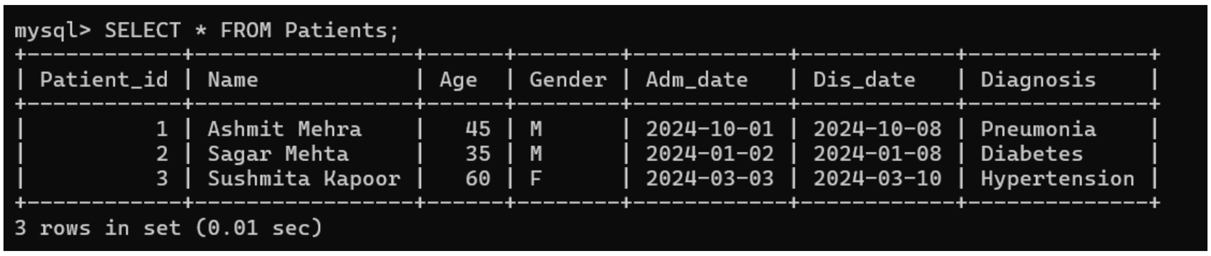
Title: Mini Project on SQL

Healthcare Analytics Database System Project

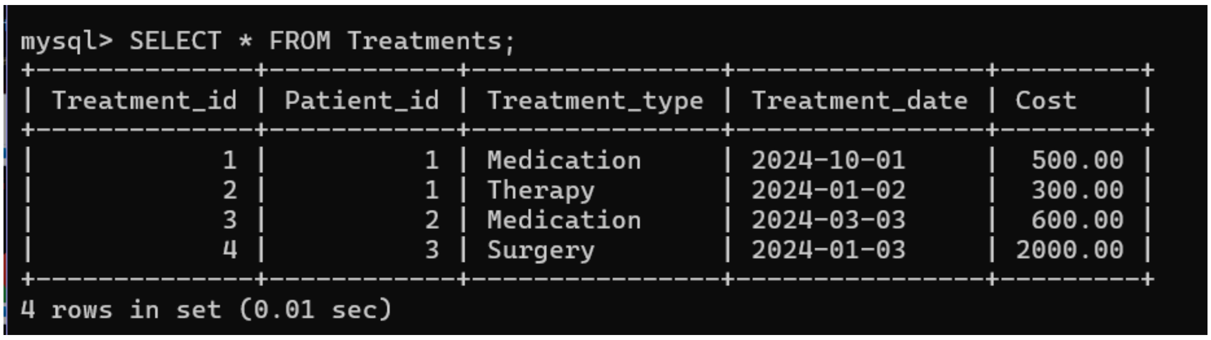
This database system is designed to store and analyse patient information, treatment details, and associated costs. It aims to provide valuable insights into patient demographics, diagnosis trends, treatment costs, and other relevant metrics.

Data Analysis and Insights:

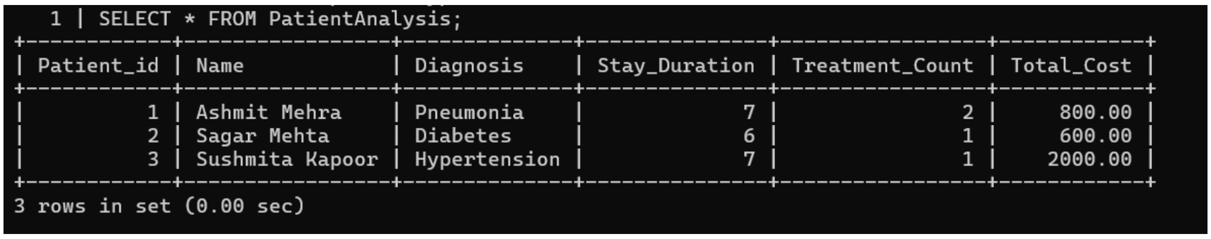
- **Patient Demographics:** Analyze patient age, gender, and distribution of diagnoses.
- **Treatment Trends:** Identify common treatments, treatment costs, and trends over time.
- **Patient Stay Duration:** Analyze the average and median stay duration for different diagnoses.
- **Cost Analysis:** Calculate the total cost of treatment per patient and per diagnosis.
- **Predictive Modeling:** Develop machine learning models to predict treatment costs, patient length of stay, or risk factors for certain diseases.



```
mysql> SELECT * FROM Patients;
+------------+-----------------+------+--------+------------+------------+--------------+
| Patient_id | Name            | Age  | Gender | Adm_date   | Dis_date   | Diagnosis    |
+------------+-----------------+------+--------+------------+------------+--------------+
|          1 | Ashmit Mehra    |   45 | M      | 2024-10-01 | 2024-10-08 | Pneumonia    |
|          2 | Sagar Mehta     |   35 | M      | 2024-01-02 | 2024-01-08 | Diabetes     |
|          3 | Sushmita Kapoor |   60 | F      | 2024-03-03 | 2024-03-10 | Hypertension |
+------------+-----------------+------+--------+------------+------------+--------------+
3 rows in set (0.01 sec)
```



```
mysql> SELECT * FROM Treatments;
+--------------+------------+----------------+----------------+---------+
| Treatment_id | Patient_id | Treatment_type | Treatment_date | Cost    |
+--------------+------------+----------------+----------------+---------+
|            1 |          1 | Medication     | 2024-10-01     |  500.00 |
|            2 |          1 | Therapy        | 2024-01-02     |  300.00 |
|            3 |          2 | Medication     | 2024-03-03     |  600.00 |
|            4 |          3 | Surgery        | 2024-01-03     | 2000.00 |
+--------------+------------+----------------+----------------+---------+
4 rows in set (0.01 sec)
```



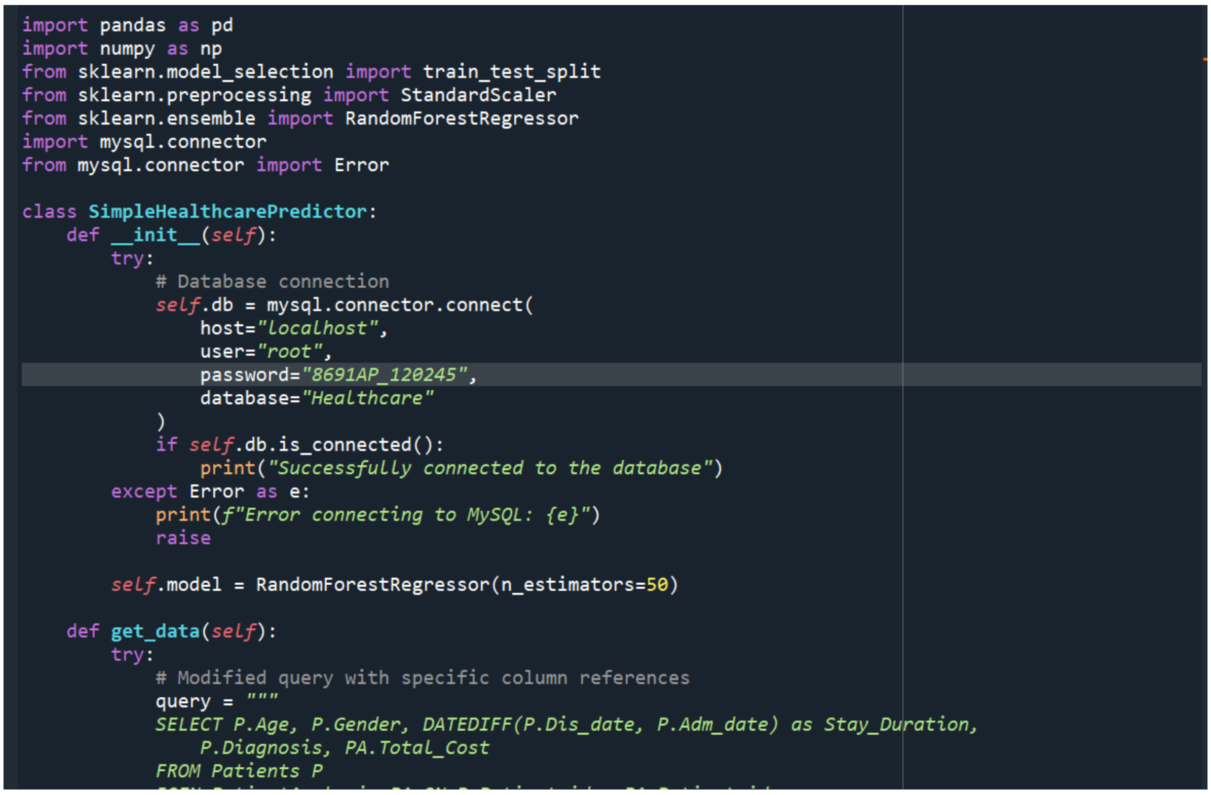
```
  1 | SELECT * FROM PatientAnalysis;
+------------+-----------------+--------------+---------------+-----------------+------------+
| Patient_id | Name            | Diagnosis    | Stay_Duration | Treatment_Count | Total_Cost |
+------------+-----------------+--------------+---------------+-----------------+------------+
|          1 | Ashmit Mehra    | Pneumonia    |             7 |               2 |     800.00 |
|          2 | Sagar Mehta     | Diabetes     |             6 |               1 |     600.00 |
|          3 | Sushmita Kapoor | Hypertension |             7 |               1 |    2000.00 |
+------------+-----------------+--------------+---------------+-----------------+------------+
3 rows in set (0.00 sec)
```

Python Code for Machine Learning Prediction

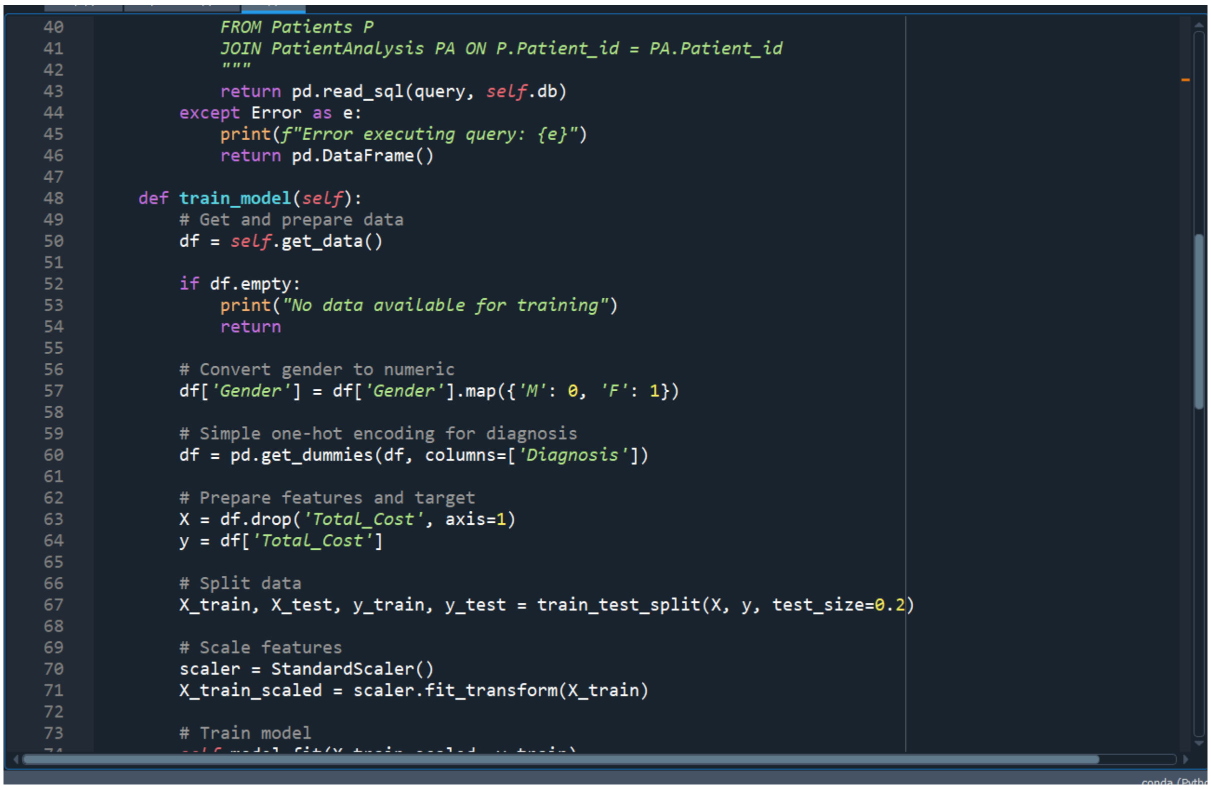
Python Implementation:

The provided Python code demonstrates how to:

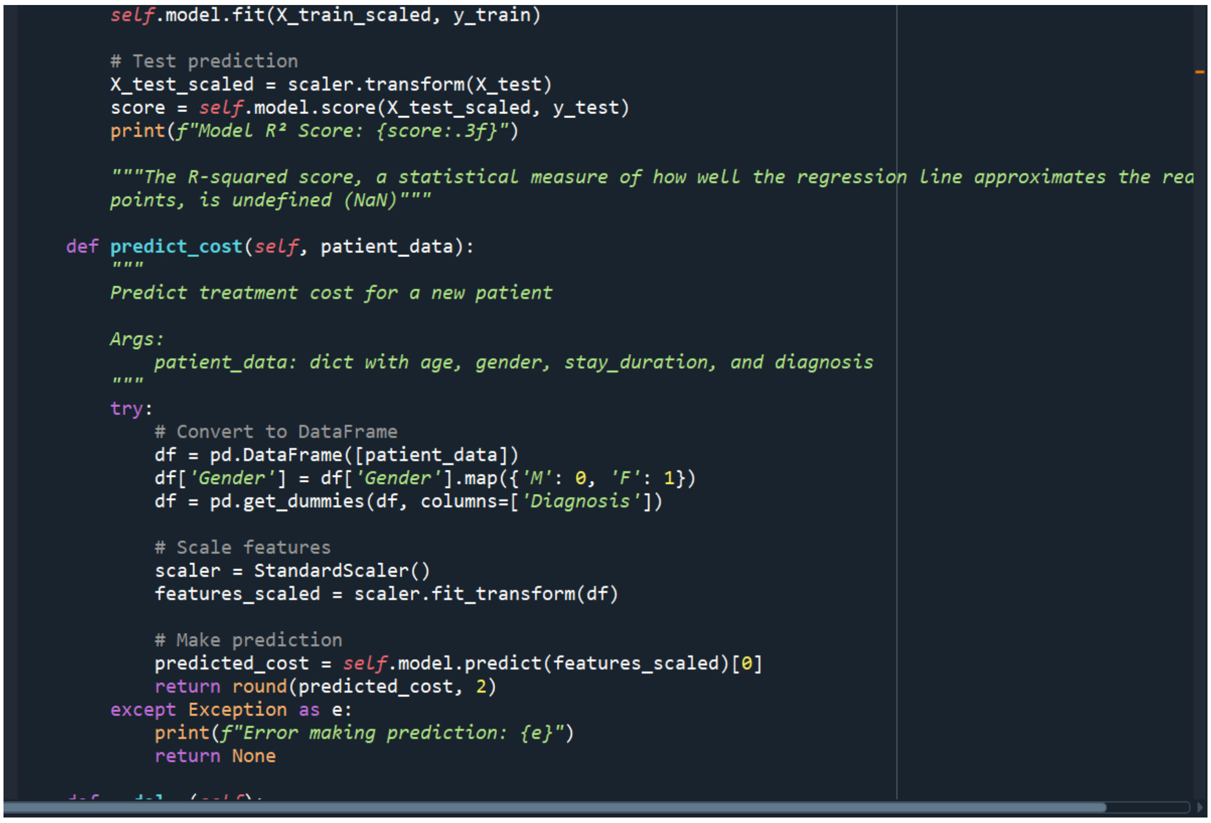
1. **Connect to the Database:** Establish a connection to the MySQL database using MySQL. Connector library.
2. **Retrieve Data:** Use SQL queries to fetch data from the database tables.
3. **Data Preprocessing:** Clean and preprocess the data, handling missing values, outliers, and categorical features.
4. **Model Training:** Train a machine learning model (e.g., Random Forest Regressor) to predict treatment costs based on patient features.
5. **Model Evaluation:** Evaluate the model's performance using R-squared and Mean Squared Error metrics.
6. **Prediction:** Use the trained model to predict treatment costs for new patients.



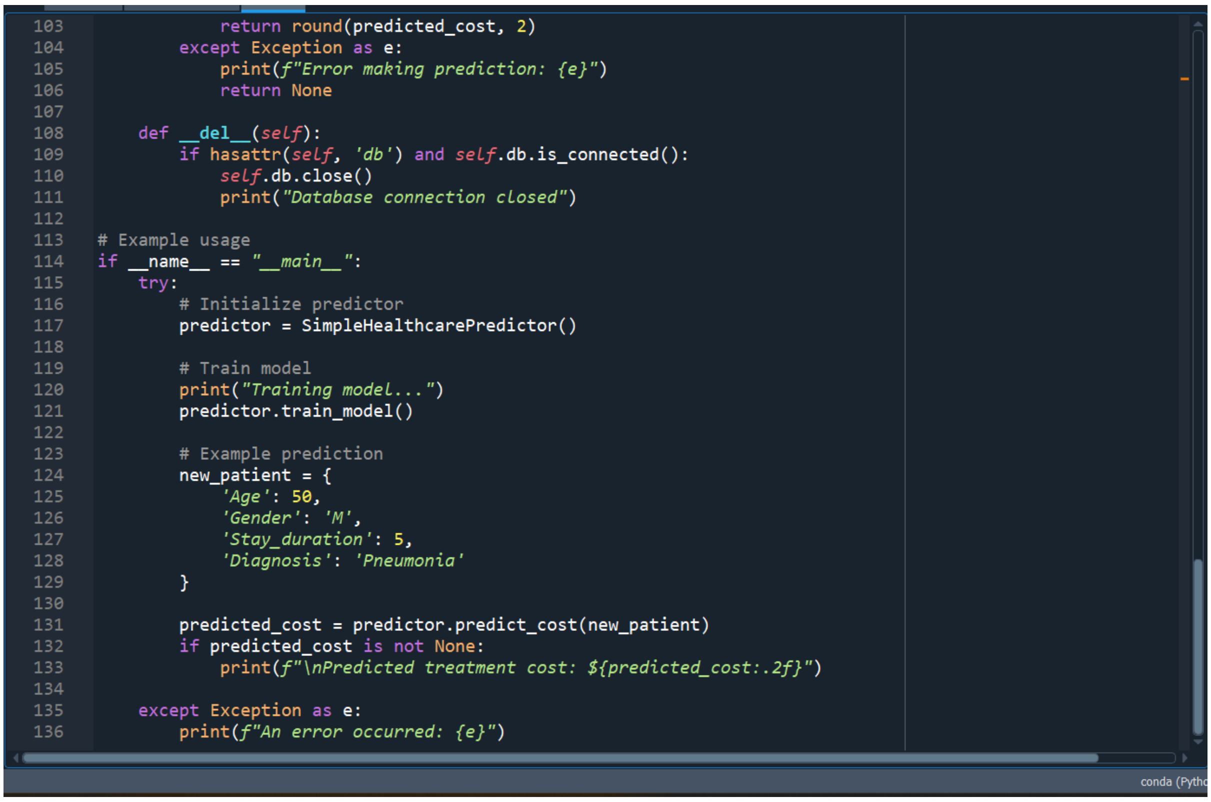
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import mysql.connector
from mysql.connector import Error

class SimpleHealthcarePredictor:
    def __init__(self):
        try:
            # Database connection
            self.db = mysql.connector.connect(
                host="localhost",
                user="root",
                password="8691AP_120245",
                database="Healthcare"
            )
            if self.db.is_connected():
                print("Successfully connected to the database")
        except Error as e:
            print(f"Error connecting to MySQL: {e}")
            raise

        self.model = RandomForestRegressor(n_estimators=50)

    def get_data(self):
        try:
            # Modified query with specific column references
            query = """
            SELECT P.Age, P.Gender, DATEDIFF(P.Dis_date, P.Adm_date) as Stay_Duration,
                P.Diagnosis, PA.Total_Cost
            FROM Patients P
```



```python
            FROM Patients P
            JOIN PatientAnalysis PA ON P.Patient_id = PA.Patient_id
            """
            return pd.read_sql(query, self.db)
        except Error as e:
            print(f"Error executing query: {e}")
            return pd.DataFrame()

    def train_model(self):
        # Get and prepare data
        df = self.get_data()

        if df.empty:
            print("No data available for training")
            return

        # Convert gender to numeric
        df['Gender'] = df['Gender'].map({'M': 0, 'F': 1})

        # Simple one-hot encoding for diagnosis
        df = pd.get_dummies(df, columns=['Diagnosis'])

        # Prepare features and target
        X = df.drop('Total_Cost', axis=1)
        y = df['Total_Cost']

        # Split data
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

        # Scale features
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)

        # Train model
```

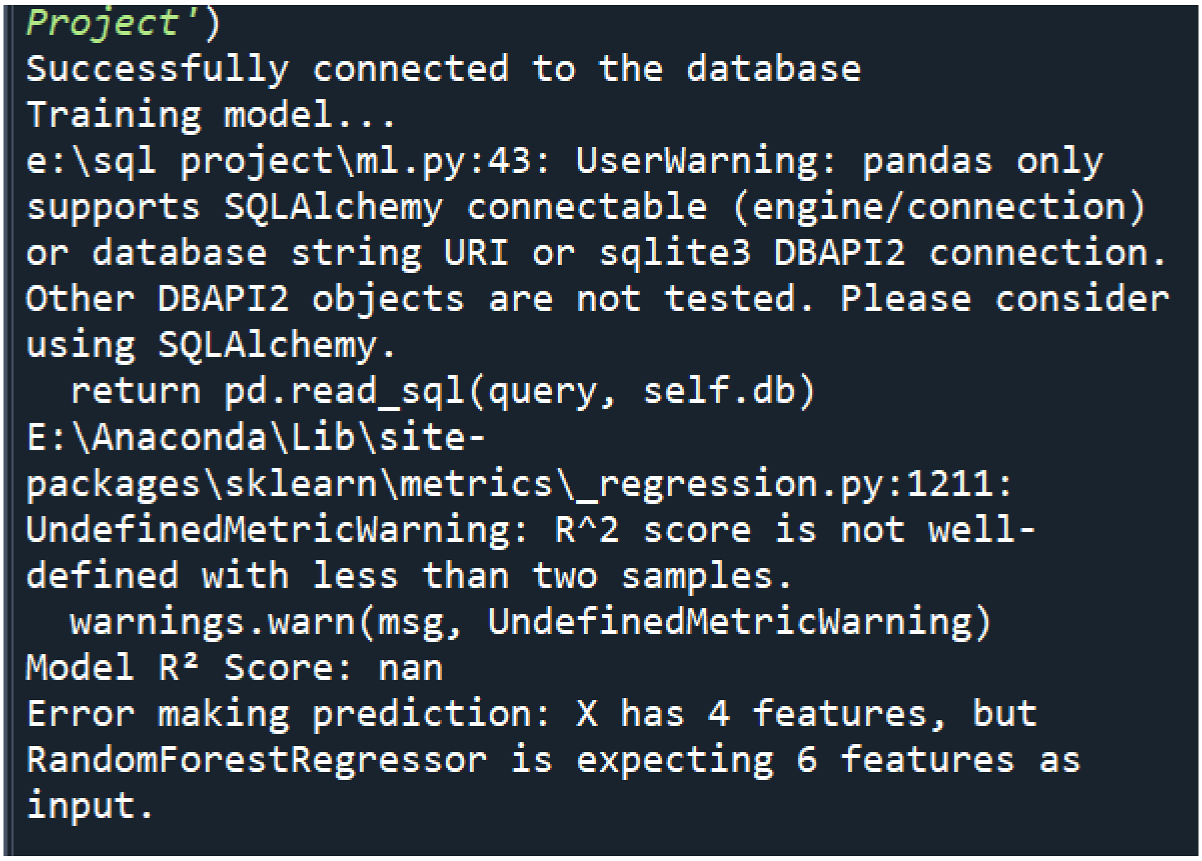


```python
        self.model.fit(X_train_scaled, y_train)

        # Test prediction
        X_test_scaled = scaler.transform(X_test)
        score = self.model.score(X_test_scaled, y_test)
        print(f"Model R² Score: {score:.3f}")

        """The R-squared score, a statistical measure of how well the regression line approximates the rea
        points, is undefined (NaN)"""

    def predict_cost(self, patient_data):
        """
        Predict treatment cost for a new patient

        Args:
            patient_data: dict with age, gender, stay_duration, and diagnosis
        """
        try:
            # Convert to DataFrame
            df = pd.DataFrame([patient_data])
            df['Gender'] = df['Gender'].map({'M': 0, 'F': 1})
            df = pd.get_dummies(df, columns=['Diagnosis'])

            # Scale features
            scaler = StandardScaler()
            features_scaled = scaler.fit_transform(df)

            # Make prediction
            predicted_cost = self.model.predict(features_scaled)[0]
            return round(predicted_cost, 2)
        except Exception as e:
            print(f"Error making prediction: {e}")
            return None
```



```python
                return round(predicted_cost, 2)
            except Exception as e:
                print(f"Error making prediction: {e}")
                return None

    def __del__(self):
        if hasattr(self, 'db') and self.db.is_connected():
            self.db.close()
            print("Database connection closed")

# Example usage
if __name__ == "__main__":
    try:
        # Initialize predictor
        predictor = SimpleHealthcarePredictor()

        # Train model
        print("Training model...")
        predictor.train_model()

        # Example prediction
        new_patient = {
            'Age': 50,
            'Gender': 'M',
            'Stay_duration': 5,
            'Diagnosis': 'Pneumonia'
        }

        predicted_cost = predictor.predict_cost(new_patient)
        if predicted_cost is not None:
            print(f"\nPredicted treatment cost: ${predicted_cost:.2f}")

    except Exception as e:
        print(f"An error occurred: {e}")
```

Output:



```
Project')
Successfully connected to the database
Training model...
e:\sql project\ml.py:43: UserWarning: pandas only
supports SQLAlchemy connectable (engine/connection)
or database string URI or sqlite3 DBAPI2 connection.
Other DBAPI2 objects are not tested. Please consider
using SQLAlchemy.
  return pd.read_sql(query, self.db)
E:\Anaconda\Lib\site-
packages\sklearn\metrics\_regression.py:1211:
UndefinedMetricWarning: R^2 score is not well-
defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
Model R² Score: nan
Error making prediction: X has 4 features, but
RandomForestRegressor is expecting 6 features as
input.
```