S GOKUL [RA2211003011996]

MUTHUMANI JD [RA2211003012002]

R LOKESHWARAN [RA2211003012018]

ASHWIN [RA2211003012009]

APPLE HEALTH

A Case Study

SOURCE CODE:

```
from google.colab import files
uploaded = files.upload()


import pandas as pd


# Replace 'sleep_data.csv' with the exact file name if different
df = pd.read_csv('sleep_data.csv')


# Show first 5 rows
df.head()
# Remove any duplicate rows
df = df.drop_duplicates()


# Strip whitespace from column names (if any)
df.columns = df.columns.str.strip()


# Show basic info
df.info()


# Check missing values
df.isnull().sum()


# Example: Fill missing values with median (modify as needed)
df.fillna(df.median(numeric_only=True), inplace=True)
```

```
# Or drop rows with missing values (if very few)
# df.dropna(inplace=True)

# Convert categorical column to category (if any)
# df['Gender'] = df['Gender'].astype('category')

# Normalize a numeric column (Min-Max Scaling)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# Apply scaling to all numeric columns
df[df.select_dtypes(include='number').columns] = scaler.fit_transform(
   df.select_dtypes(include='number'))

df.head()

# Show column names
print(df.columns)

# See the unique values in the last column (assumed target)
print("Target column:", df.columns[-1])
print("Unique values:", df[df.columns[-1]].unique())

from google.colab import files
uploaded = files.upload()

import pandas as pd

# Load the uploaded file
```

```
df = pd.read_csv(next(iter(uploaded)))

# Drop duplicates and fill missing values
df = df.drop_duplicates()
df.fillna(df.median(numeric_only=True), inplace=True)

# Convert categorical columns
df['Category'] = df['Category'].astype(str)
df['Category'] = pd.factorize(df['Category'])[0]

# Convert time columns to numeric features
df['Start Time'] = pd.to_datetime(df['Start Time'])
df['End Time'] = pd.to_datetime(df['End Time'])

df['StartHour'] = df['Start Time'].dt.hour
df['EndHour'] = df['End Time'].dt.hour
df['Duration'] = (df['End Time'] - df['Start Time']).dt.total_seconds() / 60

# Drop original time columns
df = df.drop(['Start Time', 'End Time', 'Timestamp', 'Source Name'], axis=1)

# Remove outliers
def remove_outliers_iqr(data, columns):
    for col in columns:
        Q1 = data[col].quantile(0.25)
        Q3 = data[col].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR
        data = data[(data[col] >= lower) & (data[col] <= upper)]
```

```
    return data

df = remove_outliers_iqr(df, df.select_dtypes(include='number').columns)

# Normalize numeric columns
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[df.select_dtypes(include='number').columns] = scaler.fit_transform(
    df.select_dtypes(include='number'))

# Define features and target
X = df.drop('Heart Rate', axis=1)
y = df['Heart Rate']

# Train/test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Regression models
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(),
    'Support Vector Regressor': SVR()
}
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    print(f"{name}:")
    print(f"  MSE: {mean_squared_error(y_test, preds):.4f}")
    print(f"  R2 Score: {r2_score(y_test, preds):.4f}")
    print("-" * 40)

import matplotlib.pyplot as plt
import numpy as np

# Plotting function
def plot_predictions(y_test, y_pred, title):
    plt.figure(figsize=(8, 5))
    plt.scatter(range(len(y_test)), y_test, label='Actual', alpha=0.6)
    plt.scatter(range(len(y_pred)), y_pred, label='Predicted', alpha=0.6)
    plt.title(f'{title} - Actual vs Predicted')
    plt.xlabel('Sample Index')
    plt.ylabel('Heart Rate (scaled)')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# Plot for each model
for name, model in models.items():
    preds = model.predict(X_test)
    plot_predictions(y_test.values, preds, name)
# Feature importance plot for Random Forest
import seaborn as sns
```

```
rf_model = models['Random Forest']
importances = rf_model.feature_importances_
features = X.columns

plt.figure(figsize=(8, 5))
sns.barplot(x=importances, y=features)
plt.title("Random Forest - Feature Importance")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Simulated Apple Health data
data = {
    'heart_rate': [72, 90, 85, 60, 100, 110, 65, 70, 95, 88],
    'sleep_hours': [7, 5, 6, 8, 4, 3, 7.5, 8, 5.5, 6],
    'steps_count': [8000, 4000, 3000, 10000, 2000, 1500, 8500, 9000, 2500, 3200],
    'calorie_intake': [2000, 2500, 2300, 1800, 3000, 3200, 1900, 2100, 2700, 2400],
    'health_status': ['Healthy', 'At Risk', 'At Risk', 'Healthy', 'Unhealthy', 'Unhealthy', 'Healthy', 'Healthy', 'At Risk', 'At Risk']
}
```

```
df = pd.DataFrame(data)

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

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train model
clf = RandomForestClassifier()
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Dynamically get labels in test data
unique_labels = np.unique(np.concatenate((y_test, y_pred)))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=unique_labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=unique_labels)
disp.plot(cmap='Blues')
plt.title("Apple Health Status - Confusion Matrix")
plt.show()

# Performance matrix
print("Performance Matrix:")
print(classification_report(y_test, y_pred, labels=unique_labels, target_names=unique_labels))
```

```
print(f"Overall Accuracy: {accuracy_score(y_test, y_pred):.2f}")


from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.metrics import classification_report, accuracy_score

from sklearn.model_selection import train_test_split

import pandas as pd

import numpy as np


# Simulated Apple Health data

data = {

    'heart_rate': [72, 90, 85, 60, 100, 110, 65, 70, 95, 88],

    'sleep_hours': [7, 5, 6, 8, 4, 3, 7.5, 8, 5.5, 6],

    'steps_count': [8000, 4000, 3000, 10000, 2000, 1500, 8500, 9000, 2500, 3200],

    'calorie_intake': [2000, 2500, 2300, 1800, 3000, 3200, 1900, 2100, 2700, 2400],

    'health_status': ['Healthy', 'At Risk', 'At Risk', 'Healthy', 'Unhealthy', 'Unhealthy', 'Healthy', 'Healthy', 
'At Risk', 'At Risk']

}

df = pd.DataFrame(data)


# Split features and labels

X = df.drop('health_status', axis=1)

y = df['health_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Define models

models = {

    "Decision Tree": DecisionTreeClassifier(),

    "Random Forest": RandomForestClassifier(),
```

```
    "SVM": SVC(kernel='linear', probability=True)
}

# Store performance
performance_summary = []

# Compare models
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
    performance_summary.append({
        'Model': name,
        'Accuracy': round(acc, 2),
        'Precision (avg)': round(report['weighted avg']['precision'], 2),
        'Recall (avg)': round(report['weighted avg']['recall'], 2),
        'F1-score (avg)': round(report['weighted avg']['f1-score'], 2)
    })

# Show comparison table
comparison_df = pd.DataFrame(performance_summary)
print("Model Performance Comparison:\n")
print(comparison_df)
```

OUTPUT:

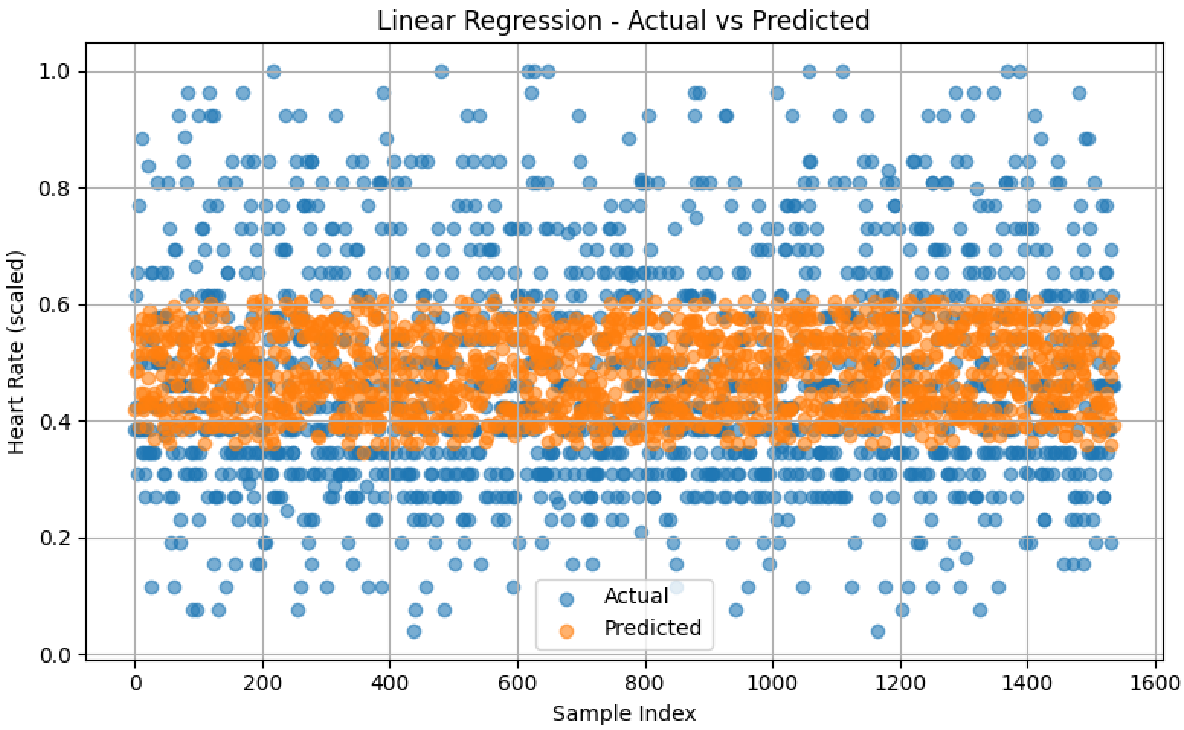
```
Model Performance Comparison:

           Model  Accuracy  Precision (avg)  Recall (avg)  F1-score (avg)
0  Decision Tree      1.00             1.00          1.00            1.00
1  Random Forest      1.00             1.00          1.00            1.00
2            SVM      0.67             0.83          0.67            0.67
```

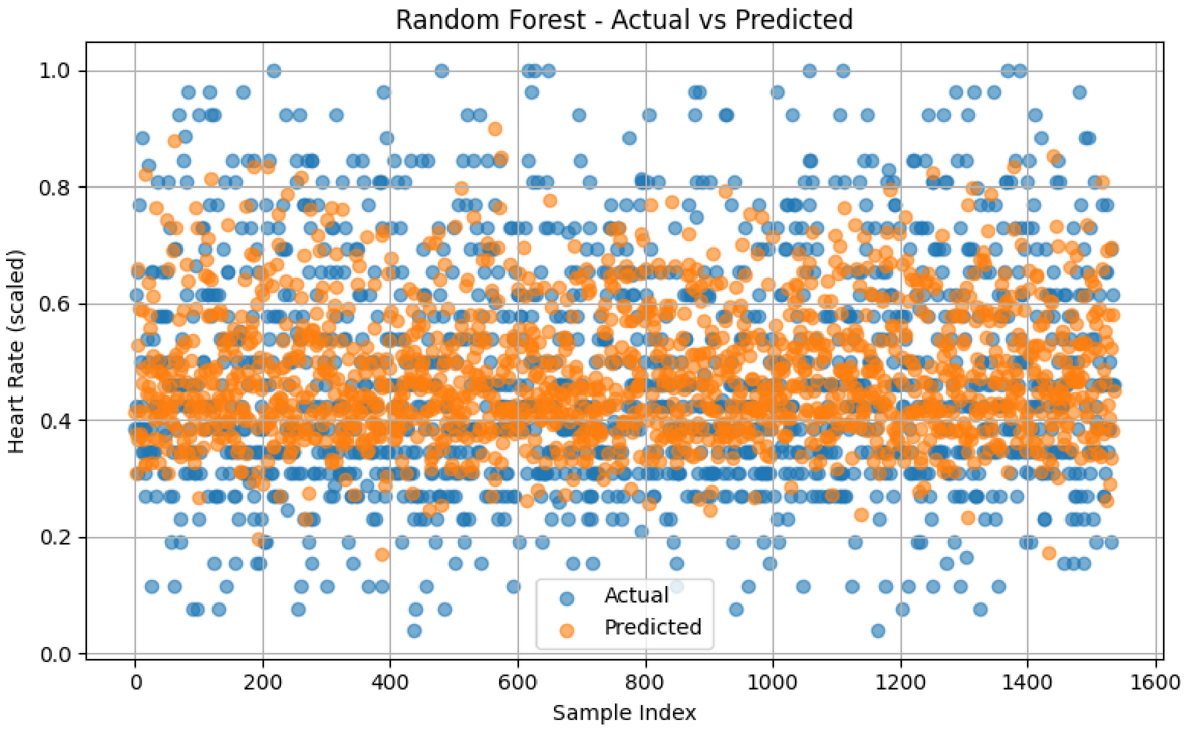
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8375 entries, 0 to 8374
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Start Time   8375 non-null   object
 1   End Time     8375 non-null   object
 2   Category     8375 non-null   object
 3   Timestamp    8375 non-null   object
 4   Heart Rate   8375 non-null   float64
 5   Source Name  8375 non-null   object
dtypes: float64(1), object(5)
memory usage: 392.7+ KB
```

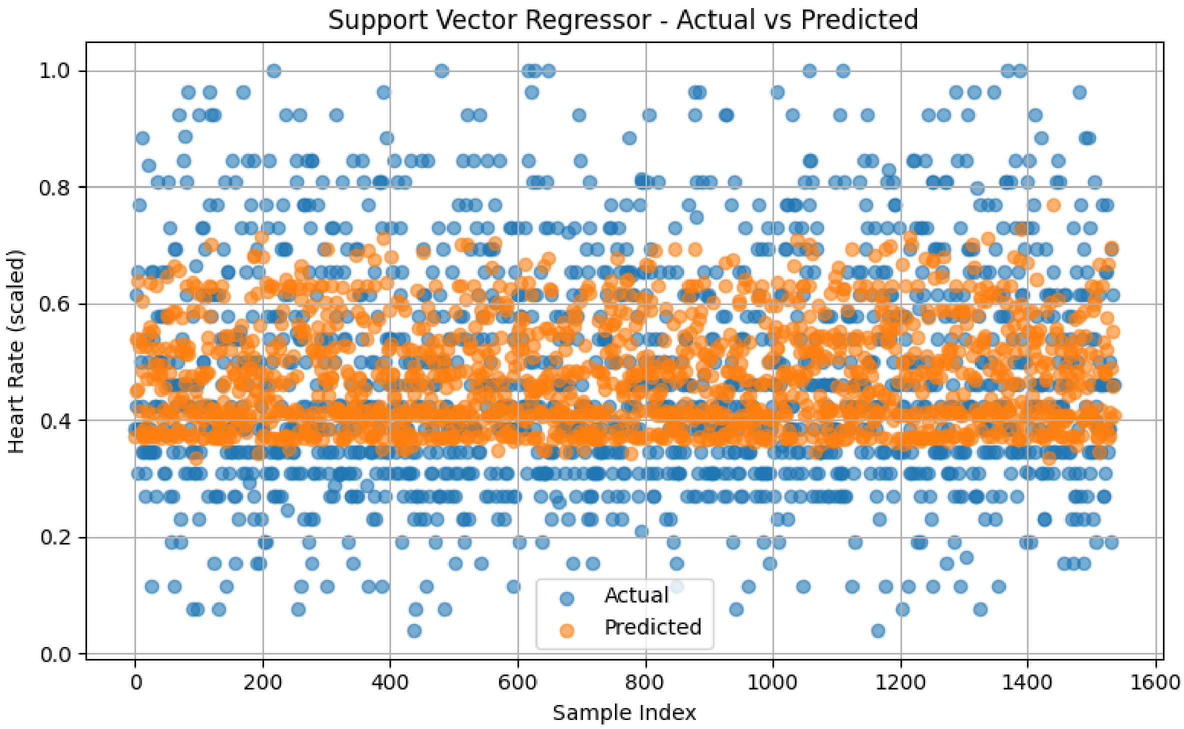
```
Choose Files  No file chosen        Upload widget is only av
Saving sleep_data.csv to sleep_data (2).csv
Linear Regression:
  MSE: 0.0325
  R2 Score: 0.1167
----------------------------------------
Random Forest:
  MSE: 0.0376
  R2 Score: -0.0236
----------------------------------------
Support Vector Regressor:
  MSE: 0.0308
  R2 Score: 0.1622
----------------------------------------
```

	Start Time	End Time	Category	Timestamp	Heart Rate	Source Name
0	2022-09-13 01:47:49	2022-09-13 01:59:19	Light/Core	2022-09-13 01:46:47	64.0000	Apple Watch SE 2020
1	2022-09-13 01:59:19	2022-09-13 02:04:49	Deep	2022-09-13 01:59:47	62.0000	Apple Watch SE 2020
2	2022-09-13 02:04:49	2022-09-13 02:12:49	Light/Core	2022-09-13 02:05:44	64.0000	Apple Watch SE 2020
3	2022-09-13 02:12:49	2022-09-13 02:27:19	Deep	2022-09-13 02:13:49	65.0000	Apple Watch SE 2020
4	2022-09-13 02:27:19	2022-09-13 02:35:49	Light/Core	2022-09-13 02:24:49	64.0000	Apple Watch SE 2020
...	...	...	...	...	...	...
8370	2023-05-14 07:20:36	2023-05-14 07:42:06	REM	2023-05-14 07:23:59	59.0000	Apple Watch Ultra
8371	2023-05-14 07:42:06	2023-05-14 07:43:06	Awake	2023-05-14 07:42:23	58.0000	Apple Watch Ultra
8372	2023-05-14 07:43:06	2023-05-14 07:45:36	Light/Core	2023-05-14 07:42:23	58.0000	Apple Watch Ultra
8373	2023-05-14 07:45:36	2023-05-14 07:52:06	REM	2023-05-14 07:45:24	62.4062	Apple Watch Ultra
8374	2023-05-14 07:52:06	2023-05-14 08:48:06	Light/Core	2023-05-14 07:51:42	63.0000	Apple Watch Ultra

8375 rows × 6 columns




Random Forest - Actual vs Predicted
Heart Rate (scaled)
Sample Index
Actual
Predicted
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
1400
1600


Support Vector Regressor - Actual vs Predicted
Heart Rate (scaled)
Sample Index
Actual
Predicted
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
1400
1600