CRISP-DM Code Review

November 1, 2024

you are a world renowned speaker and authority in CRISP-DM methodologies and have published many award winning books and articles on it. I have written a code following the CRISP-DM methodology and I would like you to critic it for me:

code:

```
import pandas as pd

# Load the dataset
data = pd.read_csv('/content/gym_members_exercise_tracking.csv')

# Display basic information about the dataset
print("Dataset Info:")
data.info()

# Display the first few rows of the dataset
print("\nFirst Few Rows:")
data.head()

# Check for missing values
print("\nMissing Values Summary:")
print(data.isnull().sum())

# Summary statistics
print("\nSummary Statistics:")
data.describe()

# Import libraries for visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of workout frequency (days per week)
plt.figure(figsize=(10, 6))
sns.countplot(data['Workout_Frequency (days/week)'])
plt.title('Distribution of Workout Frequency')
plt.xlabel('Workout Frequency (days per week)')
plt.ylabel('Count')
```

```
plt.show()

# Distribution of session duration
plt.figure(figsize=(10, 6))
sns.histplot(data['Session_Duration (hours)'], bins=20, kde=True)
plt.title('Distribution of Session Duration')
plt.xlabel('Session Duration (hours)')
plt.ylabel('Frequency')
plt.show()

# Pairplot of selected numerical features to observe relationships
selected_features = ['Age', 'Weight (kg)', 'Max_BPM', 'Avg_BPM', 'Resting_BPM', 'Session_Duration
(hours)', 'Calories_Burned']
sns.pairplot(data[selected_features])
plt.suptitle("Pairplot of Selected Numerical Features", y=1.02)
plt.show()

# Heatmap of correlations between numerical variables
plt.figure(figsize=(12, 8))
sns.heatmap(data[selected_features].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap of Selected Features")
plt.show()

from sklearn.preprocessing import StandardScaler, LabelEncoder

# Check for missing values
print("Missing Values Summary:")
print(data.isnull().sum())

# Handling Missing Values
# Dropping rows with missing values, but we could also consider filling with mean/median if
needed
data = data.dropna()

# Encoding categorical variables
# Label Encoding Gender and Workout_Type if they are categorical
label_encoder = LabelEncoder()
if data['Gender'].dtype == 'object':
    data['Gender'] = label_encoder.fit_transform(data['Gender'])

if data['Workout_Type'].dtype == 'object':
    data['Workout_Type'] = label_encoder.fit_transform(data['Workout_Type'])

# Scaling numerical features
# Selecting columns to scale
numeric_features = ['Age', 'Weight (kg)', 'Max_BPM', 'Avg_BPM', 'Resting_BPM',
                    'Session_Duration (hours)', 'Calories_Burned', 'Fat_Percentage',
                    'Water_Intake (liters)', 'Workout_Frequency (days/week)', 'BMI']
scaler = StandardScaler()
data[numeric_features] = scaler.fit_transform(data[numeric_features])
```

```
# Display the first few rows of the prepared dataset
print("\nPrepared Data Sample:")
data.head()

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Linear Regression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_pred_linreg = linreg.predict(X_test)
print("Linear Regression - Mean Squared Error:", mean_squared_error(y_test, y_pred_linreg))
print("Linear Regression - Mean Absolute Error:", mean_absolute_error(y_test, y_pred_linreg))
print("Linear Regression - R^2 Score:", r2_score(y_test, y_pred_linreg))

# Random Forest Regressor
rf_reg = RandomForestRegressor(random_state=42)
rf_reg.fit(X_train, y_train)
y_pred_rf_reg = rf_reg.predict(X_test)
print("\nRandom Forest Regressor - Mean Squared Error:", mean_squared_error(y_test,
y_pred_rf_reg))
print("Random Forest Regressor - Mean Absolute Error:", mean_absolute_error(y_test,
y_pred_rf_reg))
print("Random Forest Regressor - R^2 Score:", r2_score(y_test, y_pred_rf_reg))

# Support Vector Regressor (SVR)
svr = SVR()
svr.fit(X_train, y_train)
y_pred_svr = svr.predict(X_test)
print("\nSupport Vector Regressor - Mean Squared Error:", mean_squared_error(y_test,
y_pred_svr))
print("Support Vector Regressor - Mean Absolute Error:", mean_absolute_error(y_test, y_pred_svr))
print("Support Vector Regressor - R^2 Score:", r2_score(y_test, y_pred_svr))

import numpy as np

# Check feature importance for Random Forest Regressor
feature_importance = rf_reg.feature_importances_
features = X.columns

# Plotting feature importance
plt.figure(figsize=(10, 6))
plt.barh(features, feature_importance, color='skyblue')
plt.xlabel("Feature Importance")
plt.ylabel("Features")
plt.title("Feature Importance from Random Forest Regressor")
plt.show()

# Plotting Actual vs Predicted values for the best model
```

```
y_pred_best = y_pred_rf_reg  # Replace with the predictions of the best-performing model

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_best, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Line for perfect prediction
plt.xlabel("Actual Workout Frequency (days/week)")
plt.ylabel("Predicted Workout Frequency (days/week)")
plt.title("Actual vs Predicted Workout Frequency")
plt.show()
```

Your code is well-structured, capturing many key aspects of the CRISP-DM methodology from data understanding to modeling. Here are some detailed critiques and suggestions:

1. Business Understanding

- It's essential to define the objective explicitly before moving forward. A brief statement on why you're analyzing this dataset (e.g., to predict workout frequency based on personal health metrics) could set the direction for the data processing and modeling phases.

2. Data Understanding

- Great work in exploring the dataset with summary statistics, missing values, and basic visualizations.
- Consider including additional visualizations, such as outlier detection methods (e.g., boxplots for continuous variables) to identify anomalies or extreme values that might affect the model's performance.

3. Data Preparation

- **Missing Values Handling**: While dropping rows with missing values is sometimes viable, it could lead to information loss. Consider filling missing values with the mean/median or using more advanced imputation techniques if there is significant data loss.
- **Encoding and Scaling**: This is well done. Since scaling is applied before splitting into train and test sets, ensure it’s correctly handled in a pipeline or separated in production.

4. Modeling

- You have included multiple models, which is excellent for comparison.
- **Feature Engineering**: You might improve model performance by adding derived features (e.g., a combination of workout duration and frequency as a new feature).
- **Train/Test Split**: It seems the variables `X_train` and `y_train` aren’t defined here. Ensure that you perform a train-test split before training the models.
- **Cross-Validation**: Adding cross-validation could improve robustness by providing a more stable estimate of model performance.

5. Evaluation

- **Metrics**: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 are great choices. You may also want to report Root Mean Squared Error (RMSE) as it's often more interpretable for non-technical audiences.
- **Model Comparison**: Once you determine the best-performing model, a detailed comparison of all models with insights into their strengths and weaknesses (e.g., why the Random Forest outperforms others) could enhance your analysis.

6. Deployment and Next Steps

- Consider adding a deployment plan, even if hypothetical, describing how the model could be used in real-world scenarios.
- Explain any future improvements or additional data sources that could enrich the model.

7. Documentation

- Documenting each step with comments or brief markdown descriptions would make it easier for others to follow along, especially if this is part of a CRISP-DM report.

These suggestions align with CRISP-DM principles and would add a solid, structured approach to your analysis, enhancing interpretability and reliability. Let me know if you'd like assistance implementing any of these suggestions!