Project Description

The purpose of this project is to build a Hybrid Expert System that can classify food items as **Healthy** or **Unhealthy** using both rule-based logic and a Machine Learning model (Random Forest).

We used a nutrition dataset that contains information such as calories, protein, carbs, fat, iron, and vitamin C. The expert system first checks simple nutrition rules. If the rules cannot decide, the system uses a trained ML model to make the final prediction.

This hybrid approach provides both **explainability** (from the rules) and **accuracy** (from the ML model).

Data set is taken from kaggle, you can access through link- https://www.kaggle.com/datasets/sonalshinde123/food-nutrition-dataset-150-everyday-foods?resource=download, and next cleaned and pre processed the data

```
import pandas as pd

df = pd.read_csv("/content/Food_Nutrition_Dataset.csv")
df.head()
```

1 to 5 of 5 entries Filter

index	food_name	category	calories	protein	carbs	fat	iron	vitamin_c
0	Apple, candied	Apples	134.0	1.34	29.61	2.15	0.12	3.6
1	Apple, raw	Apples	61.0	0.17	14.8	0.15	0.03	4.6
2	Apple, dried	Dried fruits	243.0	0.93	65.89	0.32	1.4	3.9
3	Crisp, apple	Cakes and pies	215.0	2.81	30.18	9.59	1.0	0.6
4	Apple, baked	Apples	113.0	0.32	22.7	3.08	0.19	3.9

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with df | New interactive sheet

```
num_cols = ['calories', 'protein', 'carbs', 'fat', 'iron', 'vitamin_c']
df = df[num_cols]
df.head()
```

1 to 5 of 5 entries Filter

index	calories	protein	carbs	fat	iron	vitamin_c
0	134.0	1.34	29.61	2.15	0.12	3.6
1	61.0	0.17	14.8	0.15	0.03	4.6
2	243.0	0.93	65.89	0.32	1.4	3.9
3	215.0	2.81	30.18	9.59	1.0	0.6
4	113.0	0.32	22.7	3.08	0.19	3.9

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with df | New interactive sheet

```
df = df.fillna(df.median())
```

Knowledge-Based Rules (Expert System)

The rule-based system uses simple and interpretable nutrition thresholds.

The rules are:

1. If calories < 200 and fat < 10 ⟶ classify as **Healthy (Rule)**
2. If calories > 500 or fat > 25 ⟶ classify as **Unhealthy (Rule)**
3. Otherwise ⟶ the rule-based system cannot decide ⟶ **Use ML model**

These rules are designed to be easy to understand and represent domain knowledge about food healthiness.

```
def label_food(row):
    if row['calories'] <= 350 \
       and row['fat'] <= 15 \
       and row['carbs'] <= 40:
        return 1   # healthy
    else:
        return 0   # unhealthy

df['healthy'] = df.apply(label_food, axis=1)
df['healthy'].value_counts()
```

healthy	count
1	133
0	72

dtype: int64

```
X = df[num_cols]
y = df['healthy']
```

```
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, stratify=y
)
```

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import classification_report, confusion_matrix

y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.96      0.96      0.96        27

    accuracy                           0.95        41
   macro avg       0.95      0.95      0.95        41
weighted avg       0.95      0.95      0.95        41

[[13  1]
 [ 1 26]]
```

```
def rule_based(food):
    if food['calories'] < 200 and food['fat'] < 10:
        return "Healthy (Rule)"
    elif food['calories'] > 500 or food['fat'] > 25:
        return "Unhealthy (Rule)"
    else:
        return "Use ML"
```

```
sample = X_test.iloc[0]
rule = rule_based(sample)

if rule == "Use ML":
    ml_pred = model.predict([sample])[0]
    print("Hybrid Prediction:", "Healthy" if ml_pred == 1 else "Unhealthy")
else:
    print("Hybrid Prediction:", rule)
```

```
Hybrid Prediction: Unhealthy
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but
  warnings.warn(
```

```
# 1) Show train/test shapes and label balance
print("Shapes:", X_train.shape, X_test.shape)
print("Train label counts:\n", y_train.value_counts())
print("Test label counts:\n", y_test.value_counts())

# 2) Feature importances (which nutrients matter most)
import pandas as pd
fi = pd.Series(model.feature_importances_, index=X.columns).sort_values(ascending=False)
print("Feature importances:\n", fi)

# 3) Show the actual classification report & confusion matrix (already in your notebook)
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, y_pred))
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Shapes: (164, 6) (41, 6)
Train label counts:
 healthy
1    106
0     58
Name: count, dtype: int64
Test label counts:
 healthy
1    27
0    14
Name: count, dtype: int64
Feature importances:
 calories     0.467327
carbs        0.260482
fat          0.130182
protein      0.064249
iron         0.054241
vitamin_c    0.023519
dtype: float64
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.96      0.96      0.96        27

    accuracy                           0.95        41
   macro avg       0.95      0.95      0.95        41
weighted avg       0.95      0.95      0.95        41

Confusion matrix:
 [[13  1]
 [ 1 26]]
```

Conclusion

The Hybrid Expert System successfully classified food items as healthy or unhealthy using a combination of simple rules and a RandomForest machine learning model.

The RandomForest achieved high precision and recall, correctly identifying most healthy and unhealthy foods. Calories, carbs, and fat were the most important features.

The hybrid approach provided:

- Better accuracy than rules alone
- More interpretability than ML alone
- Flexible decision-making