NutriClass: Food Classification Using Nutritional Data

Project Overview

This project focuses on building a machine learning model to classify food items into multiple categories using nutritional and contextual data such as calories, proteins, fats, carbohydrates, and preparation methods.

Domain: Food & Nutrition / Machine Learning
Tech Stack: Python, Pandas, Scikit-learn, Matplotlib, Seaborn

1. Data Understanding & Exploration

In this section, the dataset is loaded and explored to understand its structure, features, and class distribution.

```
import pandas as pd

# Load dataset
df = pd.read_csv("synthetic_food_dataset_imbalanced.csv")

# First 5 rows
df.head()
```

```
     Calories    Protein        Fat      Carbs      Sugar     Fiber      Sodium  Ch
0  290.463673  14.340084  14.152608  35.266644   4.828030  1.291432  647.553389
1  212.626748   4.080908  11.621584  23.218957  16.347814  0.130303   68.572414
2  330.695408  14.326708  19.747680  29.479164   6.251137  0.794477  663.171859
3  198.070798   9.452445   5.475896  32.097878   2.984621  1.710468  300.749543
4  274.496228   6.099547  16.256002  29.756638  17.352958  1.465676  296.314958
```

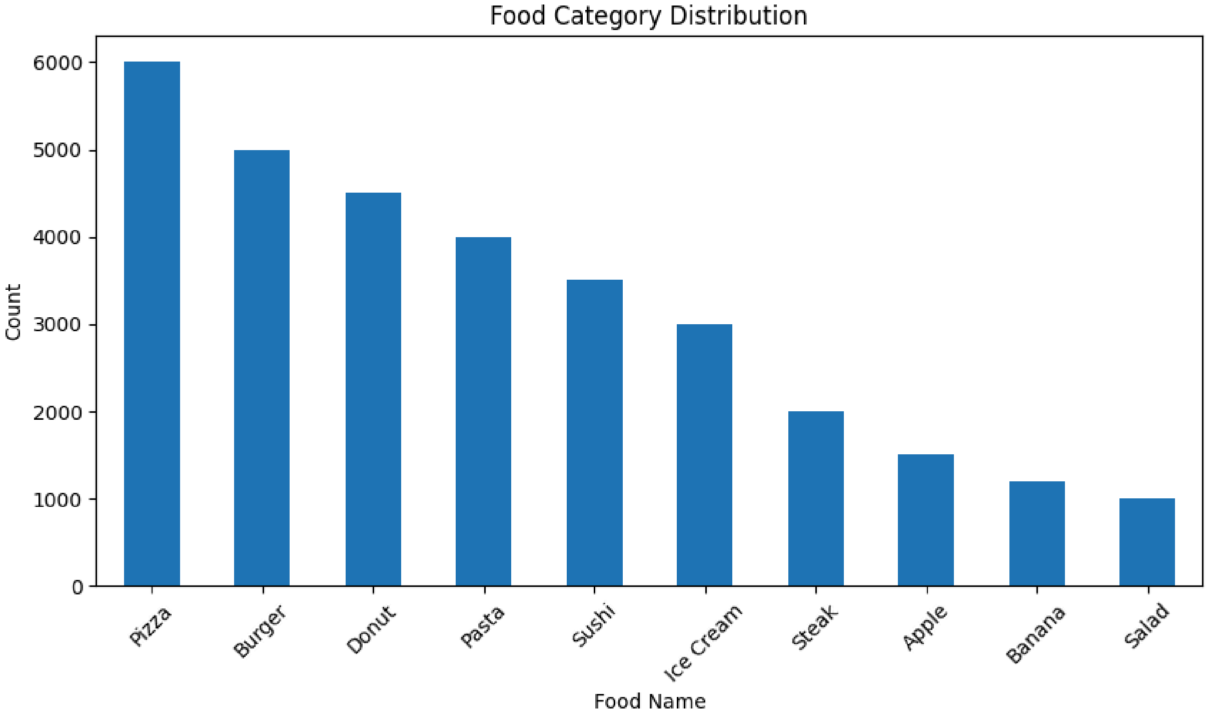
Next steps: Generate code with df New interactive sheet

```
# Dataset shape
df.shape
```

```
# Class distribution (target variable)
df['Food_Name'].value_counts()
```

```
              count
 Food_Name
     Pizza     6000
    Burger     5000
     Donut     4500
     Pasta     4000
     Sushi     3500
 Ice Cream     3000
     Steak     2000
     Apple     1500
    Banana     1200
     Salad     1000

dtype: int64
```

```
import matplotlib.pyplot as plt

df['Food_Name'].value_counts().plot(kind='bar', figsize=(10,5))
plt.title("Food Category Distribution")
plt.xlabel("Food Name")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



2. Data Preprocessing

This section includes handling missing values, removing duplicates, outlier treatment, feature encoding, and data standardization.

```
df.isnull().sum()
```

```
                      0
           Calories 375
            Protein 375
                Fat 375
              Carbs 375
              Sugar 375
              Fiber 375
             Sodium 375
        Cholesterol 375
      Glycemic_Index 375
      Water_Content 375
       Serving_Size 375
          Meal_Type   0
 Preparation_Method   0
           Is_Vegan   0
      Is_Gluten_Free  0
          Food_Name   0

dtype: int64
```

```
df.duplicated().sum()
```

```
np.int64(313)
```

```
numerical_cols = [
    'Calories','Protein','Fat','Carbs','Sugar','Fiber','Sodium',
    'Cholesterol','Glycemic_Index','Water_Content','Serving_Size'
]

categorical_cols = [
    'Meal_Type','Preparation_Method','Is_Vegan','Is_Gluten_Free'
]

target_col = 'Food_Name'
```

```
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower, upper)
```

```
df['Is_Vegan'] = df['Is_Vegan'].astype(int)
df['Is_Gluten_Free'] = df['Is_Gluten_Free'].astype(int)
```

```
df = pd.get_dummies(df, columns=['Meal_Type','Preparation_Method'], drop_first=
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df[target_col] = le.fit_transform(df[target_col])
```

```
from sklearn.preprocessing import StandardScaler

X = df.drop(columns=[target_col])
y = df[target_col]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
```

Double-click (or enter) to edit

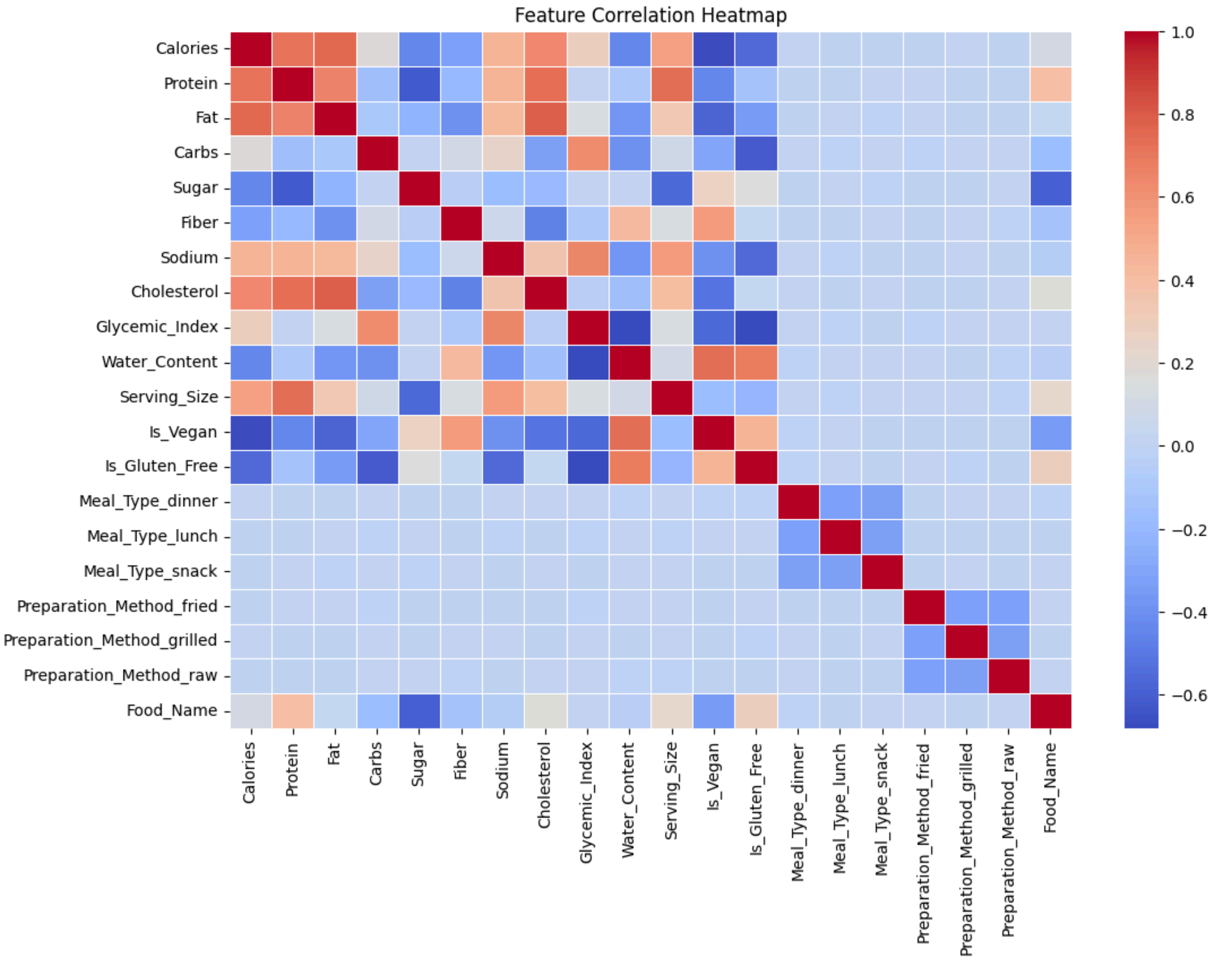
3. Feature Engineering & Selection

Feature relationships were analyzed using correlation and feature importance. Dimensionality reduction using PCA was optionally applied.

```python
import pandas as pd

# Convert scaled array back to DataFrame for analysis
X_df = pd.DataFrame(X_scaled, columns=X.columns)

# Add target back for correlation
corr_df = X_df.copy()
corr_df['Food_Name'] = y

# Correlation with target
corr_df.corr()['Food_Name'].sort_values(ascending=False)
```

```
                             Food_Name
               Food_Name      1.000000
                 Protein      0.394773
           Is_Gluten_Free     0.283305
            Serving_Size      0.225091
             Cholesterol      0.169071
                Calories      0.112461
                     Fat      0.028608
           Glycemic_Index     0.006694
          Meal_Type_snack     0.005747
  Preparation_Method_fried    0.005101
   Preparation_Method_raw     0.001892
 Preparation_Method_grilled  -0.001074
          Meal_Type_lunch    -0.003138
         Meal_Type_dinner    -0.005415
            Water_Content    -0.039902
                  Sodium     -0.050989
                   Fiber     -0.140437
                   Carbs     -0.171047
                Is_Vegan     -0.348964
                   Sugar     -0.595452

dtype: float64
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,8))
sns.heatmap(corr_df.corr(), cmap='coolwarm', linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```



```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

```
feature_importance = pd.Series(
    rf.feature_importances_, index=X.columns
).sort_values(ascending=False)

feature_importance.head(10)
```

```
                       0
        Sodium  0.196993
   Cholesterol  0.138054
  Serving_Size  0.130802
         Sugar  0.093318
 Is_Gluten_Free 0.078693
 Glycemic_Index 0.072355
       Protein  0.067986
      Calories  0.059541
           Fat  0.047170
 Water_Content  0.037777

dtype: float64
```

```
import numpy as np

np.isnan(X_scaled).sum()
```

```
np.int64(4125)
```

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')

X_imputed = imputer.fit_transform(X_scaled)
```

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95, random_state=42)
X_pca = pca.fit_transform(X_imputed)
```

```
X_pca.shape
```

```
(31700, 13)
```

```
X_final = X_imputed   # cleaned + scaled + imputed
y_final = y
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_final, y_final, test_size=0.2, random_state=42, stratify=y_final
)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

4. Model Training & Comparison

Multiple machine learning models were trained and compared using weighted evaluation metrics due to class imbalance.

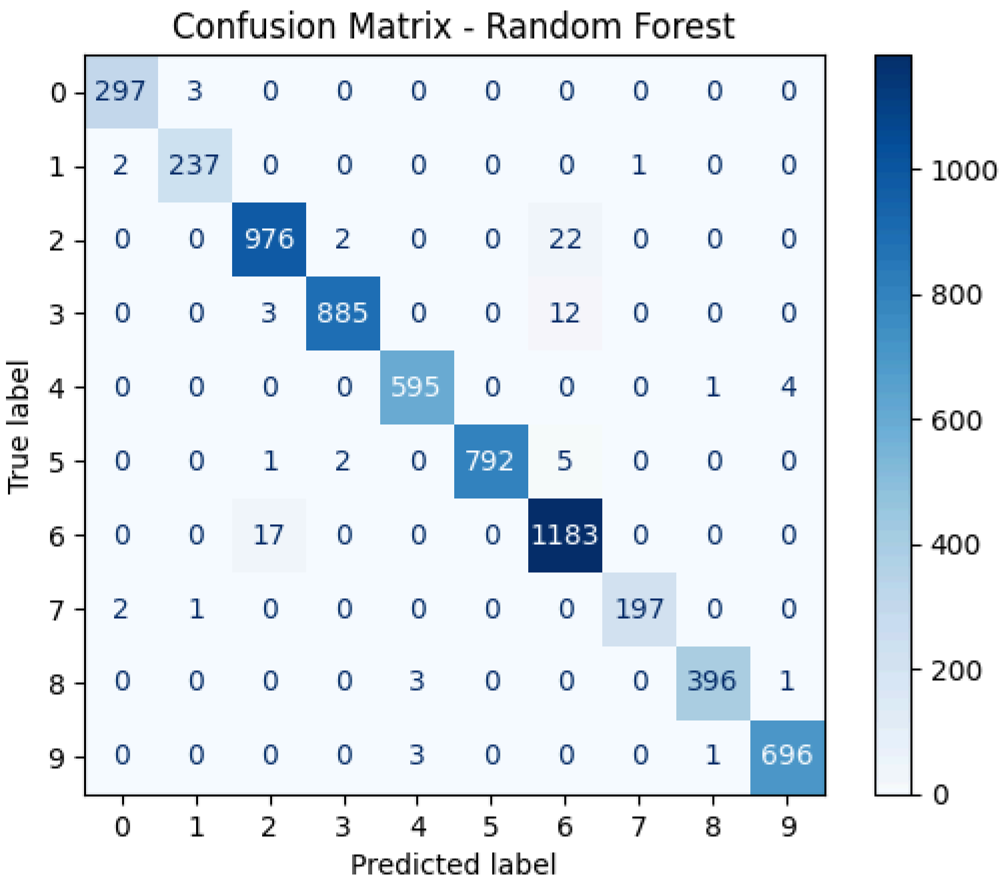
```
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s
import pandas as pd

results = []

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred, average='weighted'),
        "Recall": recall_score(y_test, y_pred, average='weighted'),
        "F1-Score": f1_score(y_test, y_pred, average='weighted')
    })

results_df = pd.DataFrame(results)
results_df
```

	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.986278	0.986489	0.986278	0.986309
1	Decision Tree	0.981230	0.981371	0.981230	0.981268
2	Random Forest	0.986435	0.986533	0.986435	0.986458
3	KNN	0.982492	0.982680	0.982492	0.982506
4	SVM	0.988170	0.988375	0.988170	0.988202
5	Gradient Boosting	0.987697	0.987896	0.987697	0.987727

Next steps: Generate code with results_df New interactive sheet

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

best_model = models["Random Forest"]
y_pred_best = best_model.predict(X_test)

cm = confusion_matrix(y_test, y_pred_best)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap="Blues")
plt.title("Confusion Matrix - Random Forest")
plt.show()
```



	0	1	2	3	4	5	6	7	8	9
0	297	3	0	0	0	0	0	0	0	0
1	2	237	0	0	0	0	0	1	0	0
2	0	0	976	2	0	0	22	0	0	0
3	0	0	3	885	0	0	12	0	0	0
4	0	0	0	0	595	0	0	0	1	4
5	0	0	1	2	0	792	5	0	0	0
6	0	0	17	0	0	0	1183	0	0	0
7	2	1	0	0	0	0	0	197	0	0
8	0	0	0	0	3	0	0	0	396	1
9	0	0	0	0	3	0	0	0	1	696

```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(
    RandomForestClassifier(random_state=42),
    X_final, y_final,
    cv=5, scoring='f1_weighted'
)

cv_scores.mean()
```

```
np.float64(0.9870424511594195)
```

```
results_df = results_df.sort_values(by="F1-Score", ascending=False).reset_index
results_df
```

	Model	Accuracy	Precision	Recall	F1-Score
0	SVM	0.988170	0.988375	0.988170	0.988202
1	Gradient Boosting	0.987697	0.987896	0.987697	0.987727
2	Random Forest	0.986435	0.986533	0.986435	0.986458
3	Logistic Regression	0.986278	0.986489	0.986278	0.986309
4	KNN	0.982492	0.982680	0.982492	0.982506
5	Decision Tree	0.981230	0.981371	0.981230	0.981268

Next steps: Generate code with `results_df` New interactive sheet

```
best_model_name = results_df.loc[0, "Model"]
best_model_name
```

```
'SVM'
```

```
best_model = models[best_model_name]
best_model.fit(X_train, y_train)
```

```
▾ SVC i ?
SVC()
```

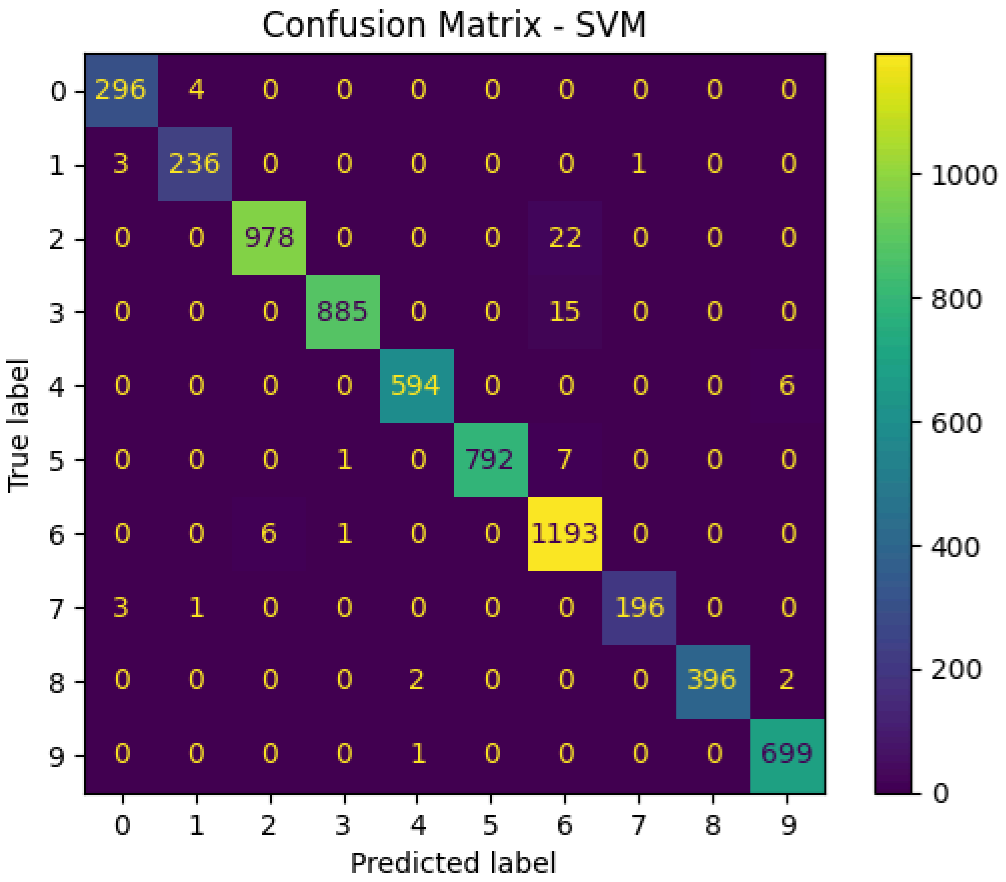
5. Evaluation & Results

Model performance was evaluated using accuracy, precision, recall, F1-score, and confusion matrix visualization.

```
from sklearn.metrics import classification_report

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       300
           1       0.98      0.98      0.98       240
           2       0.99      0.98      0.99      1000
           3       1.00      0.98      0.99       900
           4       0.99      0.99      0.99       600
           5       1.00      0.99      0.99       800
           6       0.96      0.99      0.98      1200
           7       0.99      0.98      0.99       200
           8       1.00      0.99      0.99       400
           9       0.99      1.00      0.99       700

    accuracy                           0.99      6340
   macro avg       0.99      0.99      0.99      6340
weighted avg       0.99      0.99      0.99      6340
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title(f"Confusion Matrix - {best_model_name}")
plt.show()
```



	0	1	2	3	4	5	6	7	8	9
0	296	4	0	0	0	0	0	0	0	0
1	3	236	0	0	0	0	0	1	0	0
2	0	0	978	0	0	0	22	0	0	0
3	0	0	0	885	0	0	15	0	0	0
4	0	0	0	0	594	0	0	0	0	6
5	0	0	0	1	0	792	7	0	0	0
6	0	0	6	1	0	0	1193	0	0	0
7	3	1	0	0	0	0	0	196	0	0
8	0	0	0	0	2	0	0	0	396	2
9	0	0	0	0	1	0	0	0	0	699

Double-click (or enter) to edit

```
# Display sample actual vs predicted food labels
sample_results = pd.DataFrame({
    "Actual Food": le.inverse_transform(
        y_test.iloc[:10] if hasattr(y_test, "iloc") else y_test[:10]
    ),
    "Predicted Food": le.inverse_transform(y_pred[:10])
})

sample_results
```

	Actual Food	Predicted Food
0	Steak	Steak
1	Apple	Apple
2	Pasta	Pasta
3	Pizza	Pizza
4	Sushi	Sushi
5	Pizza	Pizza
6	Pizza	Pizza
7	Donut	Donut
8	Pizza	Pizza
9	Donut	Donut

Next steps: Generate code with sample_results New interactive sheet

```
results_df = results_df.sort_values(by="F1-Score", ascending=False).reset_inde
results_df
```

	Model	Accuracy	Precision	Recall	F1-Score
0	SVM	0.988170	0.988375	0.988170	0.988202
1	Gradient Boosting	0.987697	0.987896	0.987697	0.987727
2	Random Forest	0.986435	0.986533	0.986435	0.986458
3	Logistic Regression	0.986278	0.986489	0.986278	0.986309
4	KNN	0.982492	0.982680	0.982492	0.982506
5	Decision Tree	0.981230	0.981371	0.981230	0.981268

Next steps: Generate code with results_df New interactive sheet

```
best_model_name = results_df.loc[0, "Model"]
best_model_name
```

```
'SVM'
```

```
best_model = models[best_model_name]
best_model.fit(X_train, y_train)
```

```
▾ SVC ⓘ ?
SVC()
```

```
from sklearn.metrics import classification_report

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       300
           1       0.98      0.98      0.98       240
           2       0.99      0.98      0.99      1000
           3       1.00      0.98      0.99       900
           4       0.99      0.99      0.99       600
           5       1.00      0.99      0.99       800
           6       0.96      0.99      0.98      1200
           7       0.99      0.98      0.99       200
           8       1.00      0.99      0.99       400
           9       0.99      1.00      0.99       700

    accuracy                           0.99      6340
```