**Food Expiry Days Prediction using Python – Project Report**

---

## 1. Project Overview

### 1.1 Goal

The primary objective of this project is to develop a machine learning model capable of accurately predicting the number of days a food item will remain consumable (expiry days) based on its characteristics and storage conditions.

Food items have different spoilage rates depending on their type, storage environment, packaging, and temperature. By leveraging these factors, the model aims to predict the remaining shelf life of a product before it spoils.

We focus on modeling the relationship between:

- Food-related attributes: Type of food (e.g., fruits, dairy, meat), packaging type (e.g., plastic, vacuum-sealed), storage method (e.g., refrigerator, freezer, ambient room), and temperature (°C).
- Target variable: expiry_days – the number of days before the food item becomes unsafe to consume.

This is a supervised regression problem because the target variable (expiry_days) is continuous, and the model learns patterns from labeled historical data to make predictions.

Example:
A carton of milk stored in a refrigerator at 4°C in a plastic container may have a predicted expiry of 7 days, while the same milk in a vacuum-sealed package might last 10 days. The model captures such patterns across thousands of samples to provide accurate predictions.

---

### 1.2 Business Use Case & Importance

Food spoilage is a significant global issue, impacting economy, supply chains, and the environment. According to FAO (Food and Agriculture Organization), one-third of all food produced globally is wasted, which translates to billions of dollars in losses annually and contributes heavily to greenhouse gas emissions.

By predicting expiry days accurately, businesses and consumers can take proactive actions to reduce waste and improve safety.

**Key Stakeholders & Benefits**

1. **Retailers & Supermarkets**
    - Optimize inventory management by removing items likely to spoil sooner.
    - Reduce losses from unsold expired products.
    - Plan promotions or discounts for items nearing expiry.
2. **Consumers**
    - Make informed decisions on purchasing and consumption.
    - Avoid accidental consumption of spoiled food, improving health and safety.
    - Manage home storage efficiently, reducing personal food waste.
3. **Supply Chain & Logistics**
    - Enhance cold-chain management by monitoring temperature-sensitive products.
    - Reduce spoilage during transportation and storage.
    - Optimize storage allocation based on predicted shelf life.
4. **Environmental Impact**
    - Reduce food wastage, which lowers methane emissions from decomposing food in landfills.
    - Contribute to sustainability goals and responsible consumption initiatives.

Example Scenario:
A supermarket chain can use the model to prioritize the sale of products with lower predicted expiry. For example, if two cartons of milk are available, the system may suggest selling the one with predicted 3 days remaining before the one with 7 days. This reduces wastage and maximizes profit.

---

**1.3 Key Takeaway**

Predicting food expiry is not just a technical problem; it is a business-critical and socially impactful solution.

- Technical Perspective: Involves data preprocessing, regression modeling, and feature analysis.

- Business Perspective: Reduces financial losses, enhances customer satisfaction, and improves inventory management.
- Social & Environmental Perspective: Helps reduce food wastage and associated environmental damage, contributing to sustainability initiatives.

---

**2. Dataset Module**

**2.1 Dataset Overview**

The dataset used in this project contains over 100,000 records, making it suitable for training a robust machine learning model. Each record represents a single food item stored under specific conditions along with its observed shelf life (expiry_days).

The dataset is designed to capture key factors influencing food spoilage, including both categorical and numerical attributes. This diversity in data types allows the model to learn complex interactions between variables such as storage conditions, food type, and packaging.

**Key Characteristics of the Dataset:**

- Size: 100,000+ samples
- Features: 4 input features (Food_type, Packaging, Storage, Temperature) + 1 target variable (Expiry_days)
- Purpose: Train a regression model to predict remaining shelf life of food items

Example Record:

| | Food_type | Packaging | Storage_temperature | Preservatives | Expiry_days |
|---|---|---|---|---|---|
| **0** | Snack | Vacuum | 17 | No | 203 |
| **1** | Fruit | Vacuum | 7 | No | 5 |
| **2** | Bakery | Glass | 21 | No | 4 |
| **3** | Snack | Paper | 3 | Yes | 68 |
| **4** | Vegetable | Paper | 10 | No | 2 |

This example shows how different food types and storage conditions lead to vastly different expiry days, which the model will learn to predict.

---

### 2.2 Features

| Feature | Type | Description | Example Values |
|---|---|---|---|
| Food_type | Categorical | Category of the food item | Fruits, Vegetables, Dairy, Meat, Grains |
| Packaging | Categorical | Material or method used for packaging | Plastic, Glass, Paper, Vacuum-sealed |
| Storage | Categorical | Storage environment | Refrigerator, Freezer, Room |
| Temperature | Numerical | Storage temperature in °C | 4, -18, 25 |
| Expiry_days | Target (int) | Number of days the item remains consumable | 7, 14, 30 |

---

### 2.3 Importance of Each Feature

Understanding how each feature affects food spoilage is critical for model performance.

1. **Food_type**
    - Different food categories have intrinsic lifespans.
    - Example:
        - Dairy (milk, cheese) → short shelf life (5–10 days)
        - Grains (rice, wheat) → long shelf life (30–365 days)
    - Significance: The model learns category-specific spoilage patterns.
2. **Packaging**
    - Packaging influences exposure to oxygen, moisture, and contaminants, directly affecting shelf life.
    - Example:
        - Vacuum-sealed meat → longer shelf life
        - Paper-wrapped fruits → shorter shelf life

- Significance: Enables model to differentiate between well-protected vs. exposed items.

3. **Storage**
    - Storage environment affects microbial growth and enzymatic activity.
    - Example:
        - Freezer → very slow microbial growth → extended expiry
        - Refrigerator → moderate slowdown
        - Room temperature → faster spoilage
    - Significance: Storage type is a major predictor of shelf life.
4. **Temperature**
    - Directly affects rate of chemical and biological reactions in food.
    - Lower temperatures slow down spoilage; higher temperatures accelerate it.
    - Example:
        - Milk at 4°C → lasts ~7 days
        - Milk at 25°C → lasts ~2–3 days
    - Significance: Captures fine-grained variations in shelf life even within the same storage type.

---

**3. Preprocessing Module**

Before training a machine learning model, it is crucial to clean and preprocess the dataset. Proper preprocessing ensures accuracy, stability, and robustness of the model. This module covers handling missing values, encoding categorical variables, and splitting the data for training and testing.

**3.1 Handling Missing Values**

Missing values are common in real-world datasets and, if not handled properly, can bias the model or cause errors during training.

**Strategy Used:**

1. Categorical Features (Food_type, Packaging, Storage)

- Missing values are replaced with the mode (most frequently occurring value).
- Reason: Mode preserves the most common category, which is a reasonable assumption in large datasets.

Example:

**foodtype = d["Food_type"].mode()[0]**

**d["Food_type"].fillna(foodtype, inplace=True)**

This ensures that all missing entries in Food_type are replaced with the most common food type in the dataset. Similar imputation is applied for Packaging and Storage.

2. Numerical Features (Temperature)
   - Missing values are replaced with the mean.
   - Reason: Mean imputation maintains the central tendency without introducing extreme values.

Example:

**temperature_mean = d["Temperature"].mean()**

**d["Temperature"].fillna(temperature_mean, inplace=True)**

Why This Matters:

- Prevents loss of data due to dropping rows with missing values.
- Maintains consistency across features for the Random Forest model.

---

**3.2 Encoding Categorical Variables**

Machine learning models require numerical input. Therefore, categorical features must be converted to numeric form.

Chosen Method: Label Encoding

- Definition: Assigns an integer value to each category.
- Example Mapping:
  - Food_type → Fruits=0, Vegetables=1, Dairy=2, Meat=3, Grains=4
  - Storage → Room=0, Refrigerator=1, Freezer=2

Reason for Label Encoding over One-Hot Encoding:

1. Large Dataset (100k+ samples):
    - One-Hot Encoding would create many additional columns (high-dimensional data), increasing memory usage and computation time.
2. Random Forest Compatibility:
    - Random Forest is tree-based, and it can handle integer-encoded categories effectively because splits are made based on feature thresholds, not distances.

Example Implementation:

```
from sklearn.preprocessing import LabelEncoder


le_food = LabelEncoder()

d["Food_type"] = le_food.fit_transform(d["Food_type"])


le_storage = LabelEncoder()

d["Storage"] = le_storage.fit_transform(d["Storage"])
```

Why This Matters:

- Converts categorical data into a format that the model can interpret.
- Preserves relationships without inflating feature space unnecessarily.

---

**3.3 Train-Test Split**

To evaluate the model's performance, the dataset is divided into training and testing sets.

**Strategy Used:**

- Training Set: 80% of the data (≈ 80,000 samples)
- Testing Set: 20% of the data (≈ 20,000 samples)

Example Implementation:

```
from sklearn.model_selection import train_test_split


X = d.drop("Expiry_days", axis=1)
```

**y = d["Expiry_days"]**

**X_train, X_test, y_train, y_test = train_test_split(**

**X, y, test_size=0.2, random_state=42**

**)**

Reasons for 80-20 Split:

1. Training Size: Provides enough data for the model to learn underlying patterns.
2. Testing Size: Sufficient data to evaluate generalization performance.
3. Random State: Ensures reproducibility of results.

Benefits of Proper Split:

- Prevents overfitting, as the model is evaluated on unseen data.
- Ensures a fair performance estimate on the test set.

---

**4. Model Module**

The model module defines which machine learning algorithm is chosen, why it is suitable, and how it is trained. This is a crucial part of the project as it directly affects prediction accuracy and reliability.

**4.1 Model Selection**

Chosen Model: Random Forest Regressor

Why Random Forest is Ideal for Food Expiry Prediction:

1. Handles Mixed Data Types:
    - Our dataset contains both categorical (Food_type, Packaging, Storage) and numerical (Temperature) features.
    - Random Forest can naturally handle integer-encoded categorical features alongside numerical features without requiring extensive preprocessing or scaling.
2. Robust to Outliers:
    - Food expiry data can have extreme values (e.g., long-life grains vs. perishable dairy).

- Random Forest uses ensemble averaging across multiple trees, reducing the impact of outliers on predictions.

3. Feature Importance Analysis:
    - Random Forest provides feature importance scores, allowing us to identify which factors (e.g., Temperature, Storage) have the most impact on expiry predictions.
    - This is valuable for business insights, helping retailers and consumers understand key spoilage drivers.
4. Non-linear Relationships:
    - Food spoilage is non-linear (e.g., small temperature increases may drastically reduce shelf life).
    - Random Forest captures complex, non-linear patterns better than simple linear models.
5. Low Tuning Requirements:
    - Works well even with default hyperparameters, allowing rapid prototyping.

---

**4.2 Model Training Parameters**

Key Parameters Set for Random Forest Regressor:

| Parameter | Value | Explanation |
|---|---|---|
| n_estimators | 100 | Number of decision trees in the forest. More trees improve stability but increase training time. |
| criterion | "squared_error" | Loss function for regression. Measures the mean squared error for splits. |
| random_state | 42 | Ensures reproducibility of results across multiple runs. |

---

**4.3 Model Training Workflow**

Step-by-Step Training Process:

1. Initialize the Model with specified parameters.
2. Fit the Model on the training set (X_train, y_train).

3. Make Predictions on the test set (X_test) to evaluate performance.

Example Implementation:

```
from sklearn.ensemble import RandomForestRegressor

# Initialize the Random Forest model
model = RandomForestRegressor(
  n_estimators=100,
  criterion="squared_error",
  random_state=42
)

# Train the model on the training dataset
model.fit(X_train, y_train)

# Predict on the test dataset
y_pred = model.predict(X_test)
```

Key Notes:

- The model automatically handles non-linear relationships and feature interactions.
- After training, model.feature_importances_ can be used to analyze which features influence expiry predictions the most.

---

**4.4 Why This Model Works for Our Use Case**

- Predicts continuous values (expiry days) accurately.

- Handles large datasets efficiently (100,000+ samples).
- Provides interpretability through feature importance.
- Robust across different food types and storage conditions, making it highly suitable for real-world deployment in supermarkets, warehouses, and smart refrigerators.

---

**5. Evaluation Module**

**5.1 Metrics Used**

- MAE (Mean Absolute Error): Measures average absolute difference.
- MSE (Mean Squared Error): Penalizes larger errors more.
- $R^2$ Score: Explains variance captured by the model.

**5.2 Sample Results**

- MAE: 6.21
- MSE: 96.70
- $R^2$ Score: 0.698 (~70% variance explained).

**5.3 Sample Predictions**

Input Features Example:

Food_type = Dairy

Packaging = Plastic

Storage = Refrigerator

Temperature = 4°C

Predicted Output:

Predicted Expiry Days = 14

**5.4 Interpretation**

- Model is good at general predictions.
- Average error ≈ 6 days, which is significant for short-life foods (e.g., milk, meat).
- Works better for long-life items (grains, frozen foods).

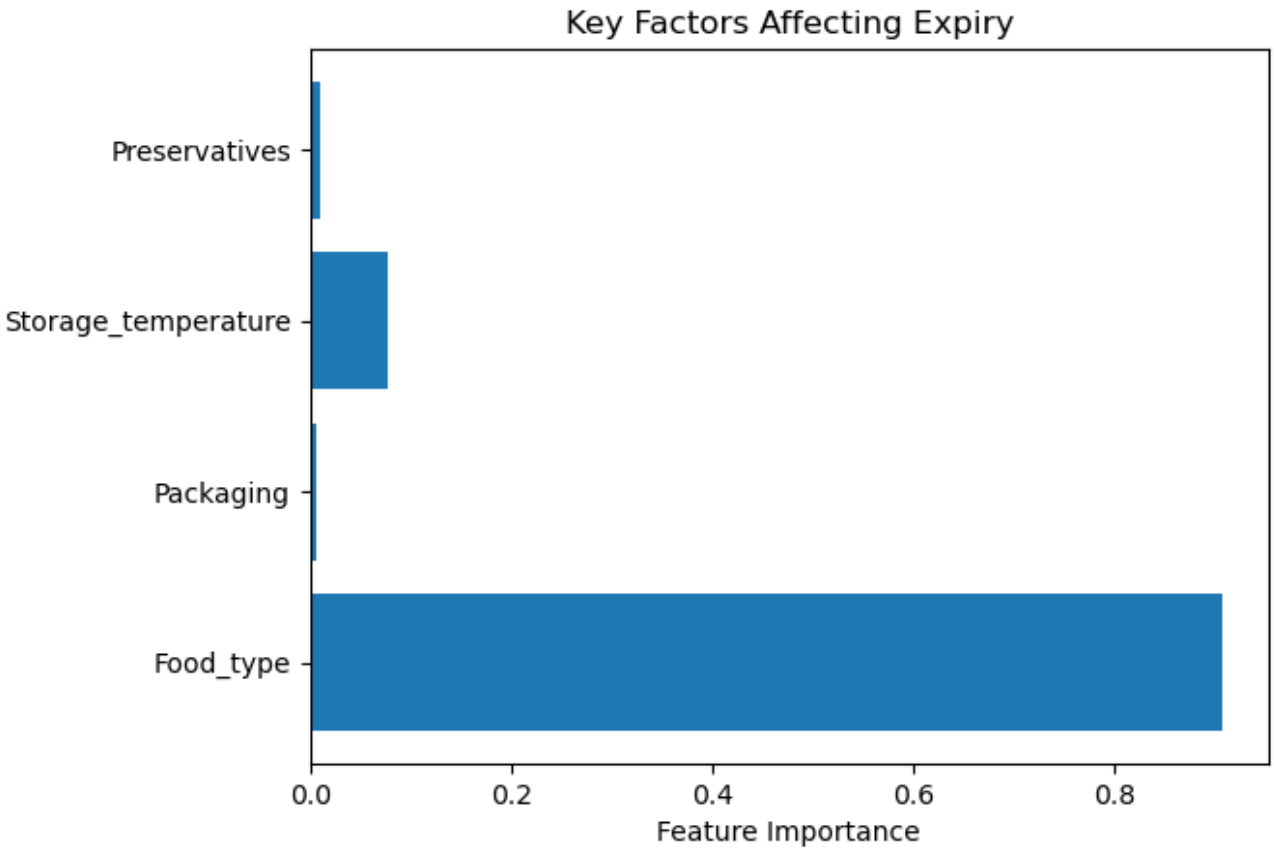
**6. Output Module**

**6.1 Prediction Output**

Given input conditions → returns predicted expiry days.

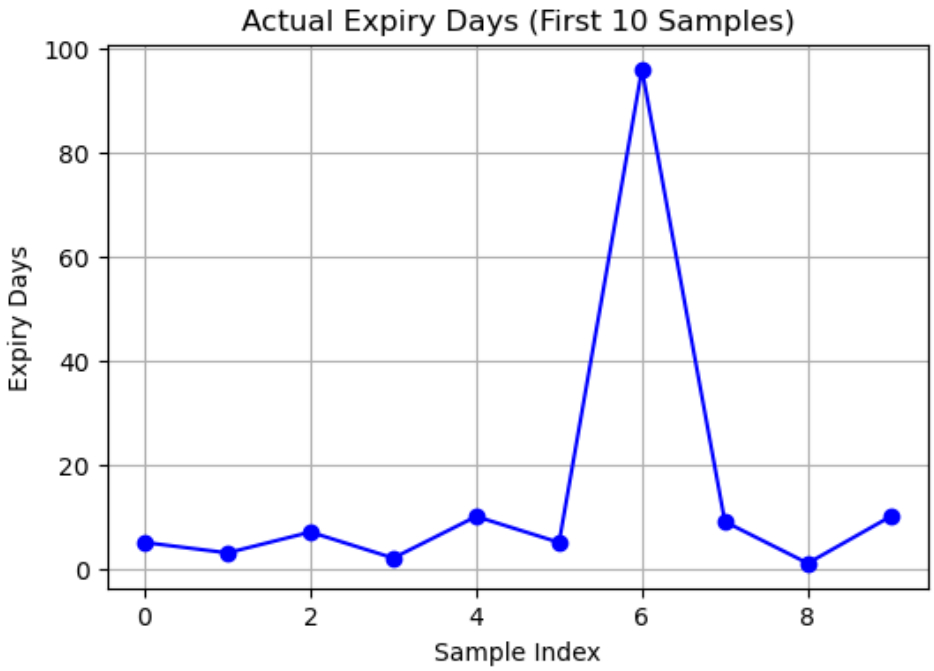
Example:

Sample Predictions = [23.36, 31.10, 7.43, 7.16, 30.72]

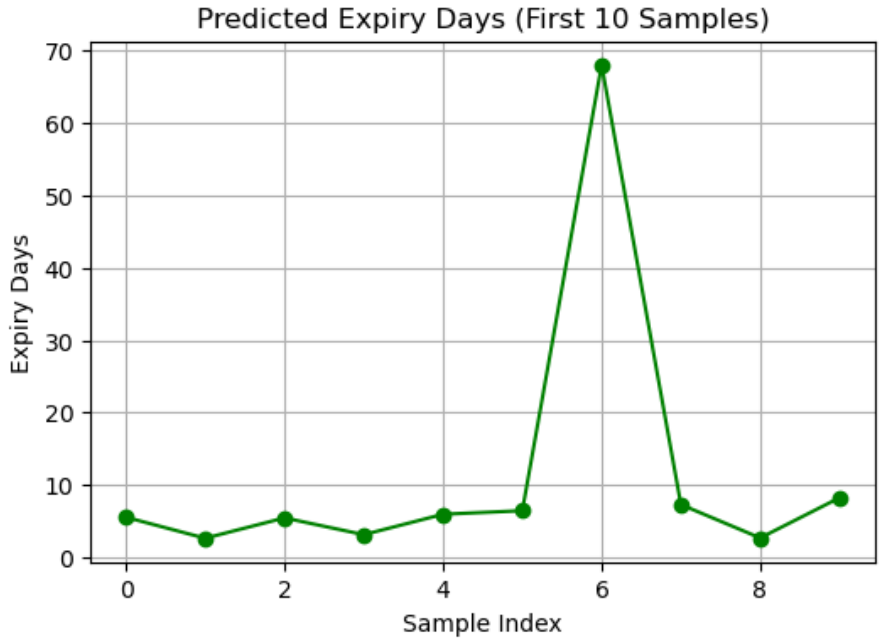



**6.2 Visualization**

- Feature Importance:
    - Temperature and Storage likely dominate expiry prediction.




- Predicted vs Actual Plot:
  - To visualize accuracy distribution.



## 7.Future Enhancements

### 7.1 Model Improvements

- Tune hyperparameters (grid search).
- Try advanced models:
    - XGBoost
    - LightGBM
    - Neural Networks

### 7.2 Feature Engineering

- Add new attributes:
    - Humidity, light exposure.
    - Nutrient profile.
    - Preservatives.

### 7.3 Practical Impact

- Smart refrigerators → automatic expiry tracking.
- Grocery store systems → automatic stock expiry alerts.
- Consumer apps → reminders for soon-to-expire items.

---