```
!pip install fpdf matplotlib seaborn
```

```
Collecting fpdf
  Downloading fpdf-1.7.2.tar.gz (39 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Building wheels for collected packages: fpdf
  Building wheel for fpdf (setup.py) ... done
  Created wheel for fpdf: filename=fpdf-1.7.2-py2.py3-none-any.whl size=40704 sha256=27143ac4ccadee7cd60e2758ea826440a6ccc8de68b4d2fdfa7
  Stored in directory: /root/.cache/pip/wheels/65/4f/66/bbda9866da446a72e206d6484cd97381cbc7859a7068541c36
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
```

```
from fpdf import FPDF
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files

# Function to generate a PDF report based on model results
def generate_report(dish_name, ingredients, risk_level, notes, alternatives, allergen_counts, risk_distribution, output_file="report.pdf"):
    """
    Generate a PDF report with the model's results and statistical analysis.

    Parameters:
        dish_name (str): Name of the dish.
        ingredients (str): List of ingredients.
        risk_level (str): Risk level (e.g., High Risk, Medium Risk).
        notes (str): Notes/warnings about allergens.
        alternatives (str): Alternative suggestions for allergens.
        allergen_counts (dict): Dictionary of allergen counts (e.g., {"Wheat": 15, "Milk": 10}).
        risk_distribution (dict): Dictionary of risk level distribution (e.g., {"High Risk": 60, "Medium Risk": 40}).
        output_file (str): Name of the output PDF file.
    """
    # Create a PDF object
    pdf = FPDF()
    pdf.add_page()

    # Set font for the title
    pdf.set_font("Arial", "B", 16)
    pdf.cell(0, 10, "Food Allergy Analysis Report", ln=True, align="C")
    pdf.ln(10)  # Add a line break

    # Add dish name
    pdf.set_font("Arial", "B", 14)
    pdf.cell(0, 10, f"Dish: {dish_name}", ln=True)
    pdf.ln(5)

    # Add ingredients
    pdf.set_font("Arial", "", 12)
    pdf.cell(0, 10, f"Ingredients: {ingredients}", ln=True)
    pdf.ln(5)

    # Add risk level
    pdf.set_font("Arial", "B", 12)
    pdf.cell(0, 10, f"Risk Level: {risk_level}", ln=True)
    pdf.ln(5)

    # Add notes
    pdf.set_font("Arial", "", 12)
    pdf.multi_cell(0, 10, f"Notes: {notes}")
    pdf.ln(5)
```

```
        # Add alternative suggestions
        pdf.set_font("Arial", "", 12)
        pdf.multi_cell(0, 10, f"Alternative Suggestions: {alternatives}")
        pdf.ln(10)

        # Add statistical analysis
        pdf.set_font("Arial", "B", 14)
        pdf.cell(0, 10, "Statistical Analysis", ln=True)
        pdf.ln(5)

        # Generate a bar chart of allergen counts
        allergens = list(allergen_counts.keys())
        counts = list(allergen_counts.values())

        plt.figure(figsize=(6, 4))
        sns.barplot(x=allergens, y=counts, palette="viridis", hue=allergens, legend=False)  # Fix Seaborn warning
        plt.title("Allergen Counts")
        plt.xlabel("Allergens")
        plt.ylabel("Count")
        plt.tight_layout()
        plt.savefig("allergen_counts.png")  # Save the chart as an image
        plt.close()

        # Add the chart to the PDF
        pdf.image("allergen_counts.png", x=10, y=pdf.get_y(), w=180)
        pdf.ln(80)  # Adjust spacing

        # Generate a pie chart of risk level distribution
        risk_levels = list(risk_distribution.keys())
        percentages = list(risk_distribution.values())

        plt.figure(figsize=(6, 4))
        plt.pie(percentages, labels=risk_levels, autopct="%1.1f%%", colors=["#ff9999", "#66b3ff"])
        plt.title("Risk Level Distribution")
        plt.tight_layout()
        plt.savefig("risk_level_distribution.png")  # Save the chart as an image
        plt.close()

        # Add the chart to the PDF
        pdf.image("risk_level_distribution.png", x=10, y=pdf.get_y(), w=180)
        pdf.ln(80)  # Adjust spacing

        # Save the PDF
        pdf.output(output_file)
        print(f"Report generated successfully: {output_file}")

    # Example usage with model results
    def get_model_results():
        """
        Simulate the model's output.
        Replace this function with actual model predictions.
        """
        # Example model results
        dish_name = "Butter Naan"
        ingredients = "Wheat, Milk, Butter, Yeast"
        risk_level = "High Risk"
        notes = "High Risk: Contains Wheat, Milk. Avoid if allergic!"
        alternatives = "Gluten-Free Flour, Rice Flour, Almond Milk, Cashew Cream, Soy Milk"

        # Example statistical data (replace with actual data from your system)
        allergen_counts = {
            "Wheat": 15,
            "Milk": 10,
            "Peanuts": 5,
            "Soy": 8,
            "Eggs": 3
        }

        risk_distribution = {
            "High Risk": 60,
            "Medium Risk": 40
        }

        return dish_name, ingredients, risk_level, notes, alternatives, allergen_counts, risk_distribution

    # Main function
    if __name__ == "__main__":
```

```
# Get model results (replace with actual model predictions)
dish_name, ingredients, risk_level, notes, alternatives, allergen_counts, risk_distribution = get_model_results()

# Generate the report
generate_report(dish_name, ingredients, risk_level, notes, alternatives, allergen_counts, risk_distribution, output_file="food_allergy_re

# Download the report
files.download("food_allergy_report.pdf")
```

```
Report generated successfully: food_allergy_report.pdf
```