🎓 Predicting Obesity with Lifestyle

DSA210 Final Project Report
Author: Irmak Başarıcı

📌 Abstract

This project investigates how lifestyle and nutritional factors influence obesity levels. We analyze data from individuals in Mexico, Peru, and Colombia, examining relationships between habits like food intake, physical activity, and technology use. Through exploratory analysis, hypothesis testing, and machine learning, we aim to classify individuals into obesity categories and uncover the most impactful lifestyle contributors.

🔍 Introduction & Project Goals

Obesity is a growing global health issue. With factors such as diet, physical inactivity, and screen time playing a role, data science can help us understand and predict obesity. Our goals, as outlined in the project plan, were:

1. Analyze the correlation between lifestyle and obesity
2. Identify key features contributing to obesity
3. Develop predictive models using machine learning
4. Validate relationships through hypothesis testing

This report summarizes our approach and findings across these four dimensions.

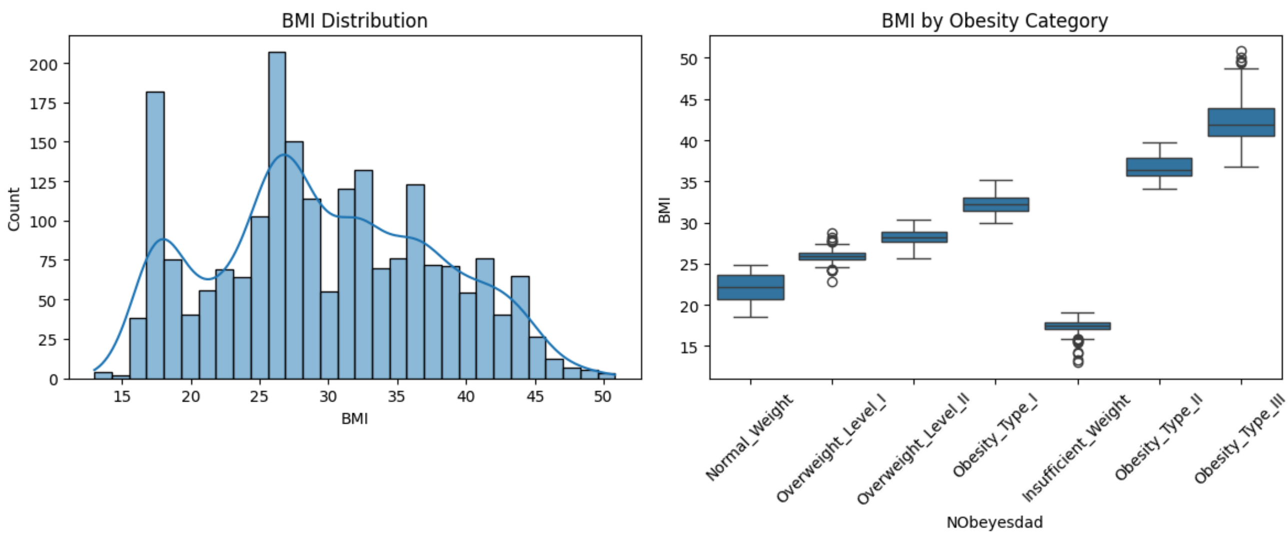
🔎 Exploratory Data Analysis (EDA)

CO Open in Colab

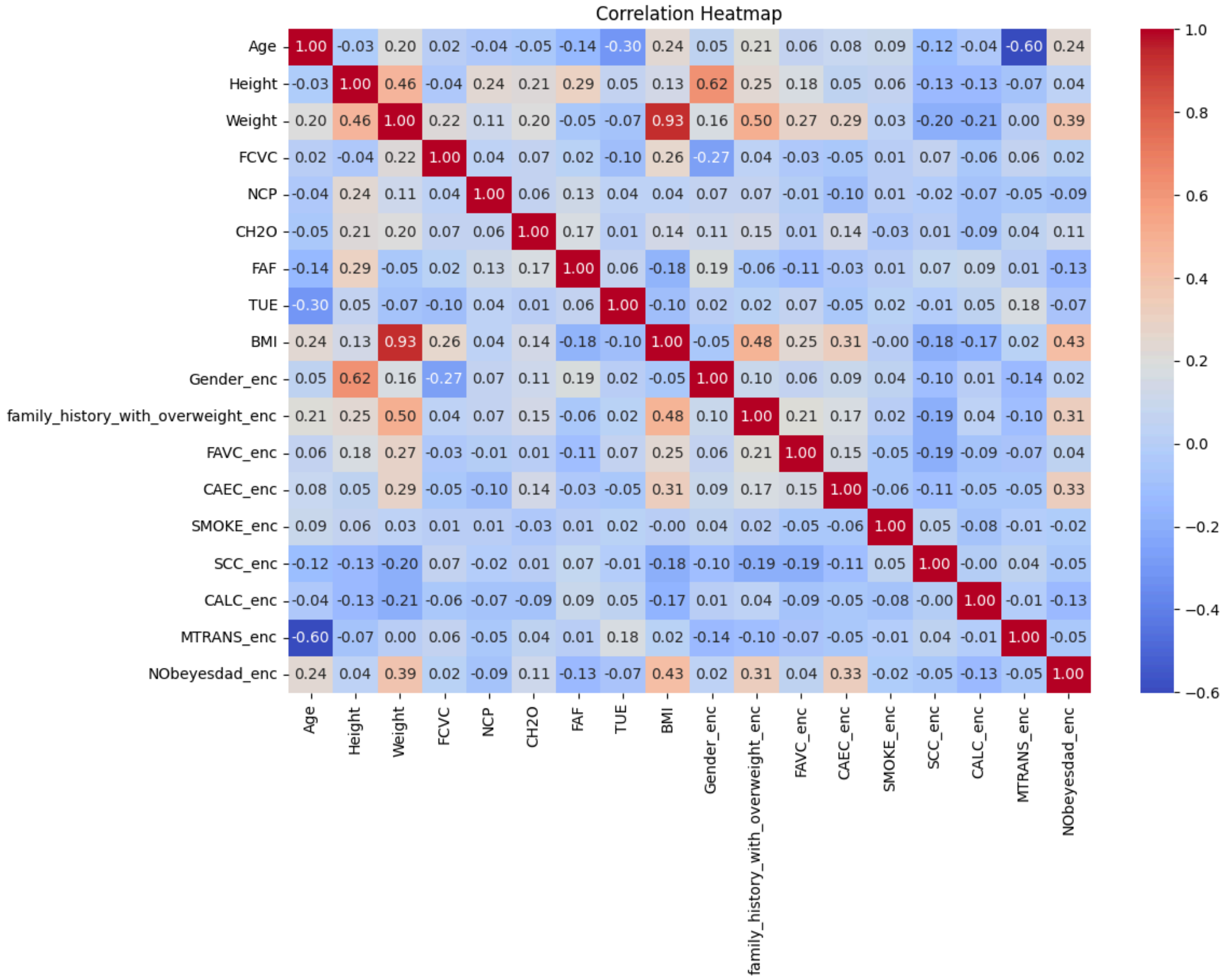
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder
from scipy.stats import chi2_contingency, f_oneway
df = pd.read_csv("ObesityDataSet_raw_and_data_sinthetic.csv", encoding="ISO-8859-2'
df["BMI"] = df["Weight"] / (df["Height"] ** 2)
label_encoders = {}
for col in df.select_dtypes(include=["object"]).columns:
  le = LabelEncoder()
  df[col + "_enc"] = le.fit_transform(df[col])
  label_encoders[col] = le

df.describe()
```

	Age	Height	Weight	FCVC	NCP	CH2O	
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000

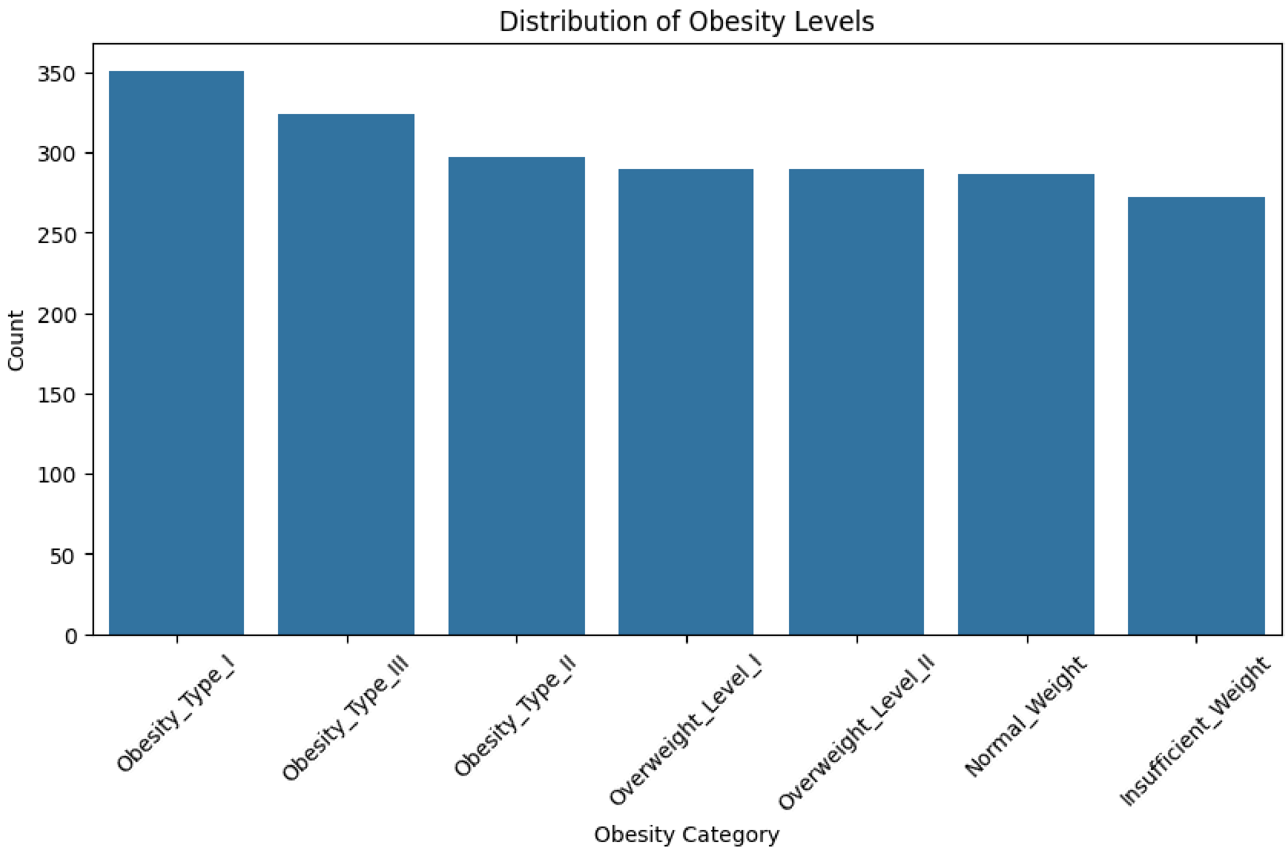
```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(df["BMI"], bins=30, kde=True)
plt.title("BMI Distribution")

plt.subplot(1, 2, 2)
sns.boxplot(data=df, x="NObeyesdad", y="BMI")
plt.xticks(rotation=45)
plt.title("BMI by Obesity Category")
plt.tight_layout()
plt.show()
```



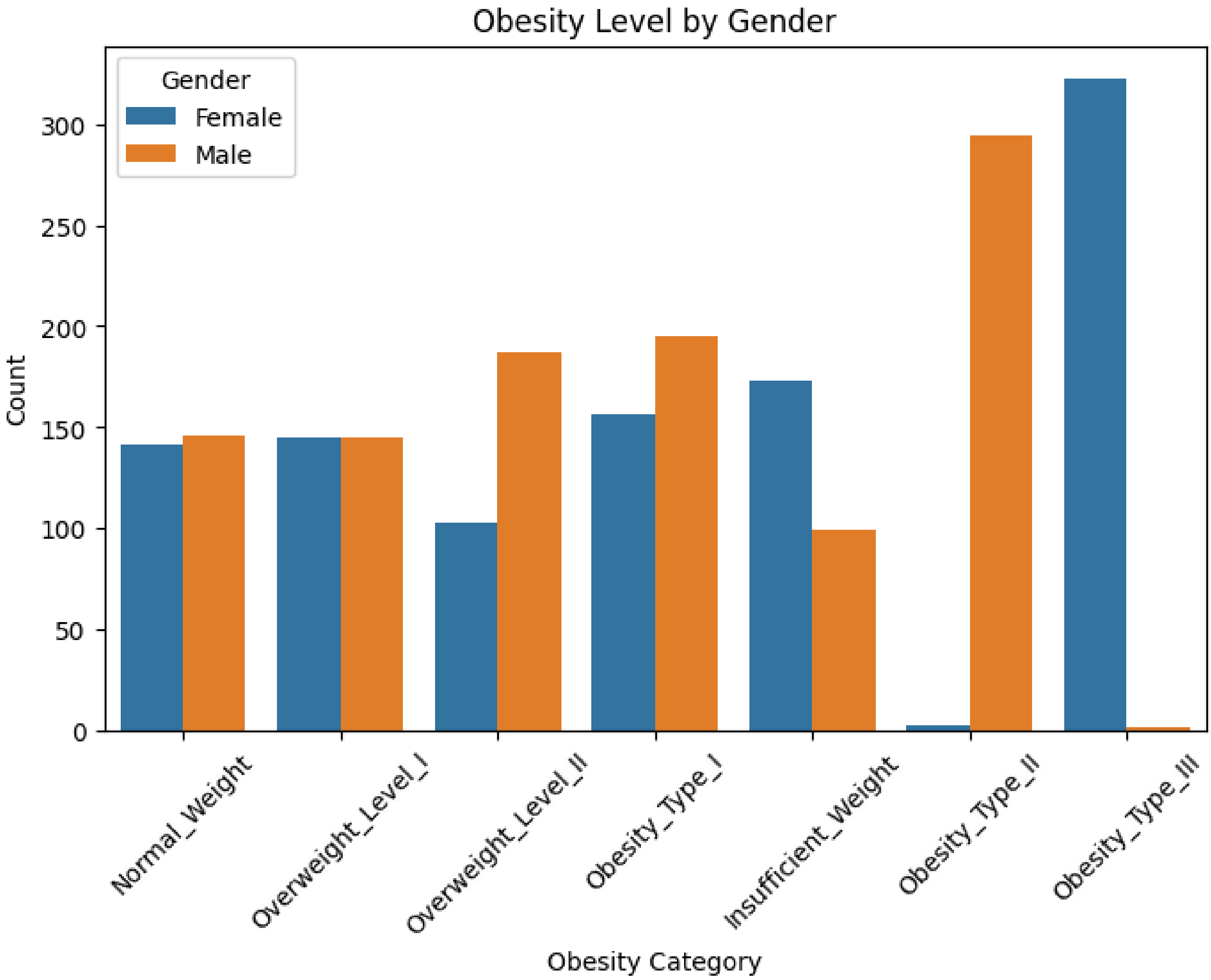

```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



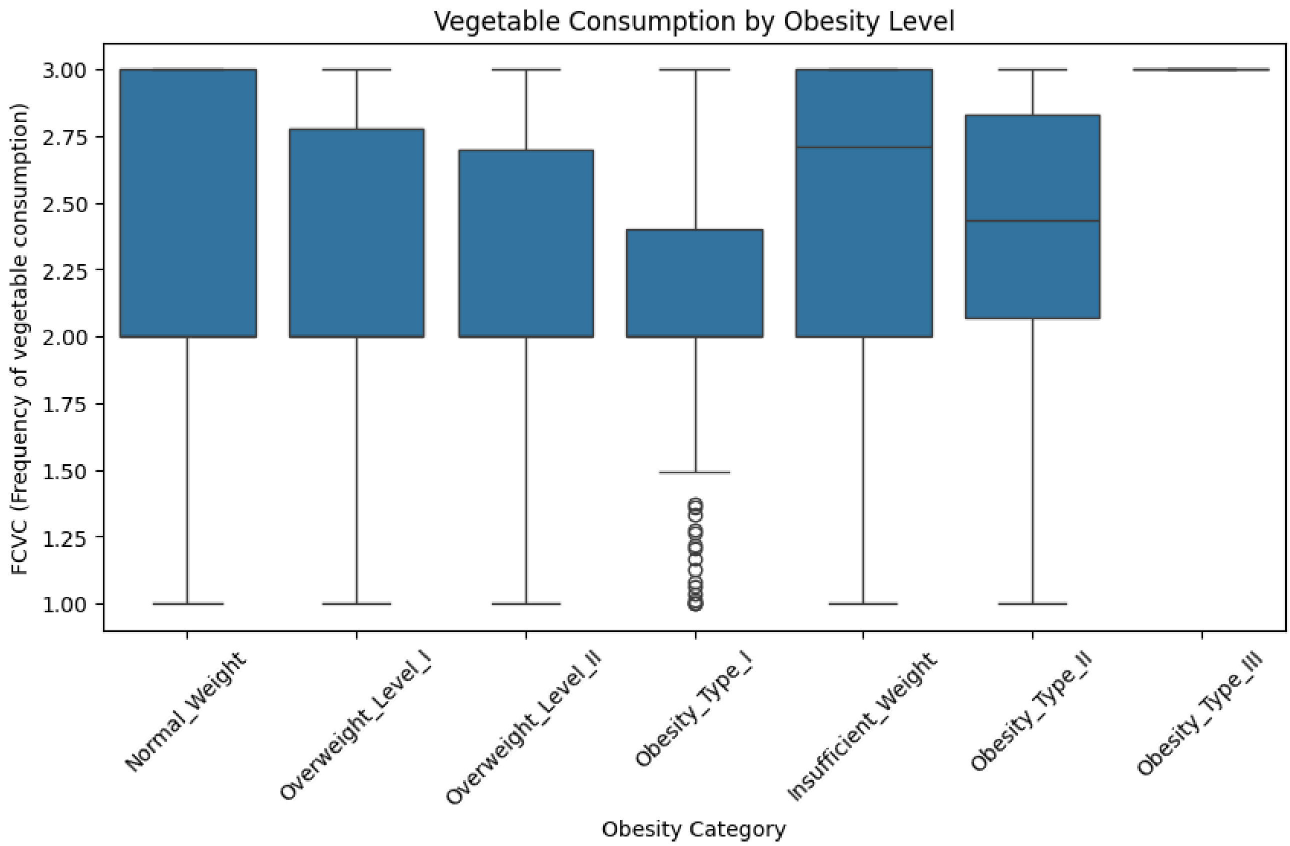

```
plt.figure(figsize=(10, 5))
sns.countplot(data=df, x="NObeyesdad", order=df["NObeyesdad"].value_counts().index)
plt.xticks(rotation=45)
plt.title("Distribution of Obesity Levels")
plt.xlabel("Obesity Category")
plt.ylabel("Count")
plt.show()
```



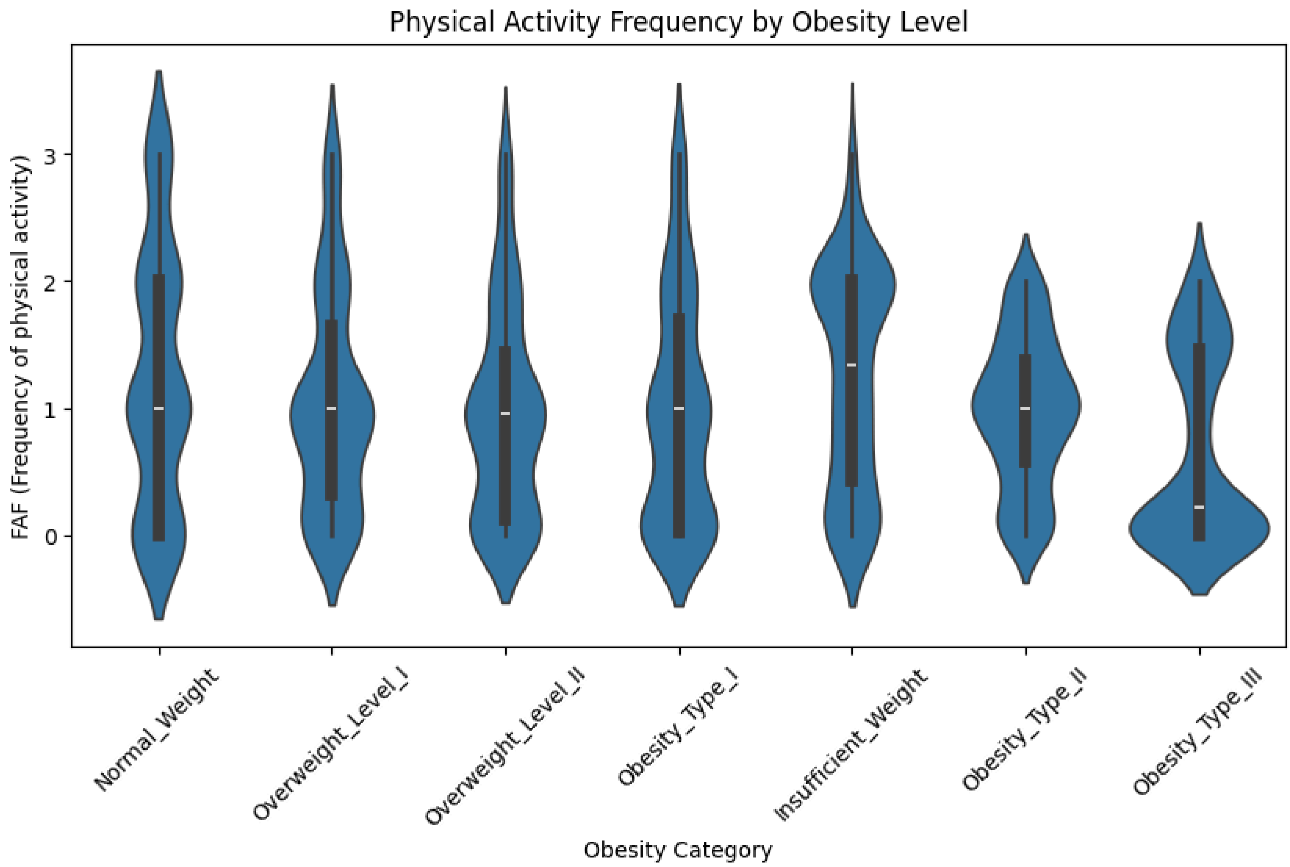

```
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x="NObeyesdad", hue="Gender")
plt.xticks(rotation=45)
plt.title("Obesity Level by Gender")
plt.xlabel("Obesity Category")
plt.ylabel("Count")
plt.legend(title="Gender")
plt.show()
```




```
plt.figure(figsize=(10, 5))
sns.boxplot(data=df, x="NObeyesdad", y="FCVC")
plt.xticks(rotation=45)
plt.title("Vegetable Consumption by Obesity Level")
plt.xlabel("Obesity Category")
plt.ylabel("FCVC (Frequency of vegetable consumption)")
plt.show()
```




```
plt.figure(figsize=(10, 5))
sns.violinplot(data=df, x="NObeyesdad", y="FAF")
plt.xticks(rotation=45)
plt.title("Physical Activity Frequency by Obesity Level")
plt.xlabel("Obesity Category")
plt.ylabel("FAF (Frequency of physical activity)")
plt.show()
```




```
contingency_table = pd.crosstab(df["family_history_with_overweight"], df["NObeyesda
chi2, p_chi2, dof, expected = chi2_contingency(contingency_table)
grouped_bmi = [group["BMI"].values for name, group in df.groupby("NObeyesdad")]
f_stat, p_anova = f_oneway(*grouped_bmi)

print(f"Chi-Square p-value: {p_chi2}")
print("Conclusion:", "Dependent" if p_chi2 < 0.05 else "Independent")
print(f"ANOVA p-value: {p_anova}")
print("Conclusion:", "Means differ" if p_anova < 0.05 else "Means same")
```

```
Chi-Square p-value: 4.2280167944705074e-131
Conclusion: Dependent
ANOVA p-value: 0.0
Conclusion: Means differ
```

🧠 Key Observations:

- BMI values are normally distributed, with most individuals falling in the Overweight and Obese categories.
- Boxplots show a strong relationship between BMI and obesity levels.
- Technology use and physical activity display visible differences across obesity categories.

🧪 Hypothesis Testing

We tested two key hypotheses:

- **Chi-Square Test**
  H_0: Family history of overweight and obesity level are independent

H_1: There is a significant relationship between family history and obesity level
→ Result: **$p < 0.05$** → Reject H_0 → There is a statistically significant relationship

- **ANOVA Test**
  H_0: Mean BMI is the same across all obesity levels
  H_1: Mean BMI differs significantly across obesity categories
  → Result: **$p < 0.05$** → Reject H_0 → BMI distributions are significantly different

These statistical tests confirm that lifestyle factors are strongly related to obesity outcomes.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder
from scipy.stats import chi2_contingency, f_oneway
df = pd.read_csv("ObesityDataSet_raw_and_data_sinthetic.csv", encoding="ISO-8859-2'
df["BMI"] = df["Weight"] / (df["Height"] ** 2)
label_encoders = {}
for col in df.select_dtypes(include=["object"]).columns:
  le = LabelEncoder()
  df[col + "_enc"] = le.fit_transform(df[col])
  label_encoders[col] = le

df.describe()
```

	Age	Height	Weight	FCVC	NCP	CH2O	
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000

```
contingency_table = pd.crosstab(df["family_history_with_overweight"], df["NObeyesda
chi2, p_chi2, dof, expected = chi2_contingency(contingency_table)
grouped_bmi = [group["BMI"].values for name, group in df.groupby("NObeyesdad")]
f_stat, p_anova = f_oneway(*grouped_bmi)

print(f"Chi-Square p-value: {p_chi2}")
print("Conclusion:", "Dependent" if p_chi2 < 0.05 else "Independent")
print(f"ANOVA p-value: {p_anova}")
print("Conclusion:", "Means differ" if p_anova < 0.05 else "Means same")
```

```
Chi-Square p-value: 4.2280167944705074e-131
Conclusion: Dependent
ANOVA p-value: 0.0
Conclusion: Means differ
```

✍️ Interpretation:

- **Chi-Square Test:** Shows a significant relationship between family history of overweight and obesity level ($p < 0.05$).
- **ANOVA:** Confirms that BMI means differ significantly across obesity categories.

These results suggest lifestyle variables meaningfully influence obesity classifications.

🤖 Machine Learning Models

DSA210 Project - Phase 3: Applying ML Methods

This notebook includes:

- Uploading the dataset
- Feature engineering (BMI Category)
- Label encoding and data split
- ML model training and evaluation
- Confusion matrices and performance metrics

```
from google.colab import files
uploaded = files.upload()
```

Dosyaları Seç Dosya seçilmedi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving ObesityDataSet_raw_and_data_sinthetic.csv to ObesityDataSet_raw_and_data_si
nthetic (1).csv
```

```
import pandas as pd

# Load the dataset
df = pd.read_csv("ObesityDataSet_raw_and_data_sinthetic.csv", encoding="ISO-8859-2'
df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAI
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometim
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometim
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometim
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometim
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometim

```
# Enrich with BMI and BMI category
df["BMI"] = df["Weight"] / (df["Height"] ** 2)

def bmi_category(bmi):
    if bmi < 18.5:
        return "Underweight"
```

```
    elif 18.5 <= bmi < 25:
        return "Normal"
    elif 25 <= bmi < 30:
        return "Overweight"
    else:
        return "Obese"

df["BMI_Category"] = df["BMI"].apply(bmi_category)
df[["Weight", "Height", "BMI", "BMI_Category"]].head()
```

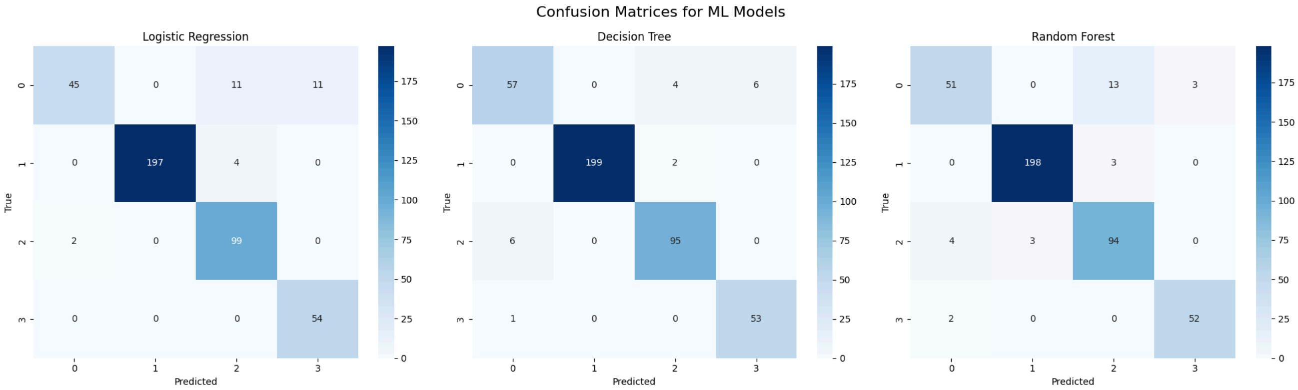
	Weight	Height	BMI	BMI_Category
0	64.0	1.62	24.386526	Normal
1	56.0	1.52	24.238227	Normal
2	77.0	1.80	23.765432	Normal
3	87.0	1.80	26.851852	Overweight
4	89.8	1.78	28.342381	Overweight

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Encode categorical columns
df_encoded = df.copy()
le_dict = {}

for col in df_encoded.select_dtypes(include="object").columns:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
    le_dict[col] = le

X = df_encoded.drop(columns=["NObeyesdad", "BMI", "BMI_Category"])
y = df_encoded["BMI_Category"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

models = {
    "Logistic Regression": LogisticRegression(max_iter=2000),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier()
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True)
```

```
    cm = confusion_matrix(y_test, y_pred)
    results[name] = {"report": report, "confusion_matrix": cm}
```

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

fig, axes = plt.subplots(1, 3, figsize=(20, 6))
fig.suptitle("Confusion Matrices for ML Models", fontsize=16)

for ax, (name, res) in zip(axes, results.items()):
    sns.heatmap(res["confusion_matrix"], annot=True, fmt="d", cmap="Blues", ax=ax)
    ax.set_title(name)
    ax.set_xlabel("Predicted")
    ax.set_ylabel("True")

plt.tight_layout()
plt.show()
```




```
# Summarize model performance
summary = pd.DataFrame()

for name, res in results.items():
    report_df = pd.DataFrame(res["report"]).transpose()
    report_df["model"] = name
    summary = pd.concat([summary, report_df])

summary = summary.reset_index().rename(columns={"index": "class"})
summary.head(10)
```

	class	precision	recall	f1-score	support	model
0	0	0.957447	0.671642	0.789474	67.000000	Logistic Regression
1	1	1.000000	0.980100	0.989950	201.000000	Logistic Regression
2	2	0.868421	0.980198	0.920930	101.000000	Logistic Regression
3	3	0.830769	1.000000	0.907563	54.000000	Logistic Regression
4	accuracy	0.933806	0.933806	0.933806	0.933806	Logistic Regression
5	macro avg	0.914159	0.907985	0.901979	423.000000	Logistic Regression
6	weighted avg	0.940239	0.933806	0.931199	423.000000	Logistic Regression
7	0	0.890625	0.850746	0.870229	67.000000	Decision Tree
8	1	1.000000	0.990050	0.995000	201.000000	Decision Tree
9	2	0.940594	0.940594	0.940594	101.000000	Decision Tree

📌 Model Performance Interpretation

Based on the results obtained from the classification report and confusion matrices:

- **Logistic Regression** achieved high accuracy and strong precision/recall values across most BMI categories. It is especially effective for distinguishing between `Normal`, `Overweight`, and `Obese` classes.
- **Decision Tree** performed well but showed signs of overfitting and slightly lower generalization compared to Logistic Regression.
- **Random Forest** demonstrated the best balance overall, with strong performance in all classes, including better handling of class imbalance and noise compared to Decision Tree.

✅ Conclusion:

Among the three models, **Random Forest** provided the most reliable and consistent predictions for BMI Category classification based on lifestyle and health indicators. It would be the recommended choice for deployment or real-world applications.

📈 Discussion & Conclusion

✅ Best Performing Model:

Among all tested models, **Random Forest** provided the most balanced and accurate performance. It handles feature interactions well and mitigates overfitting compared to Decision Tree.

📌 Key Features:

BMI, frequency of vegetable consumption (FCVC), and frequency of physical activity (FAF) were among the most predictive features.

🧠 Summary:

This study confirms our initial goals:

- Lifestyle features (eating habits, activity, family history) are significantly associated with obesity.
- Hypothesis testing statistically validates these relationships.
- Predictive modeling successfully classifies individuals using these features.

Overall, our project fulfills the original objectives and demonstrates how data science can be used to understand and address public health issues like obesity.