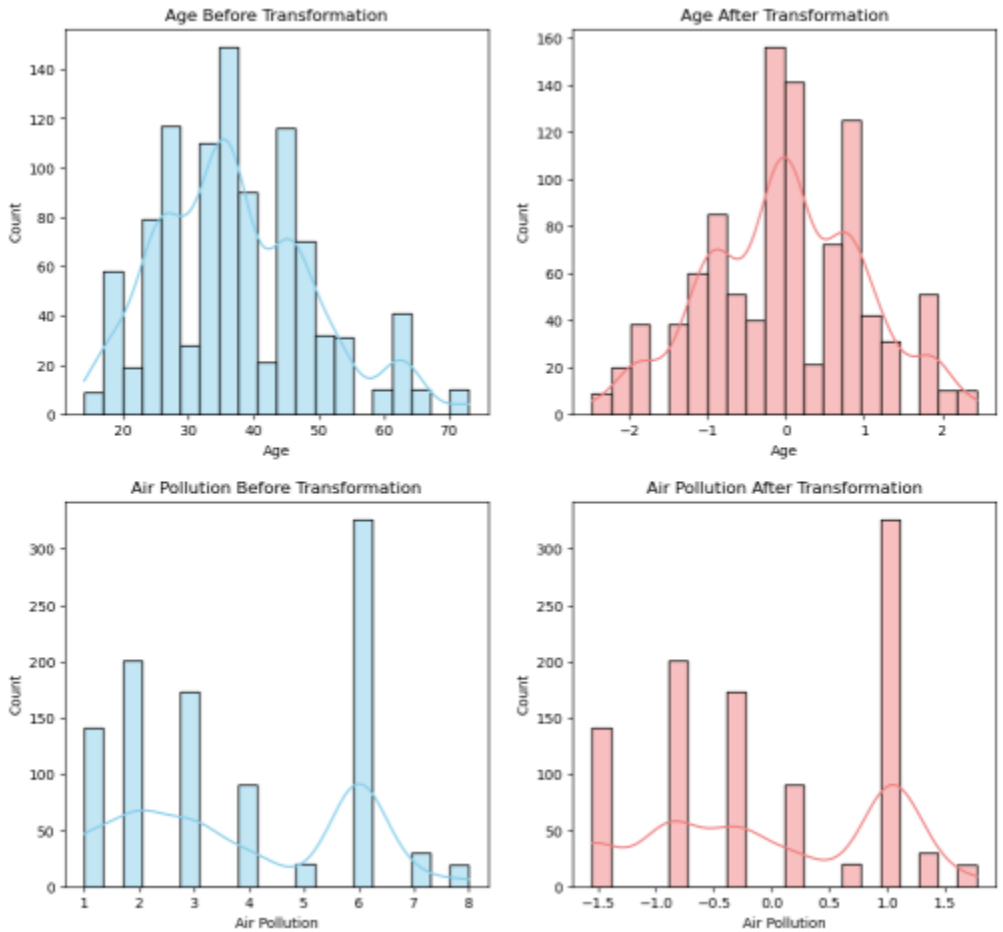
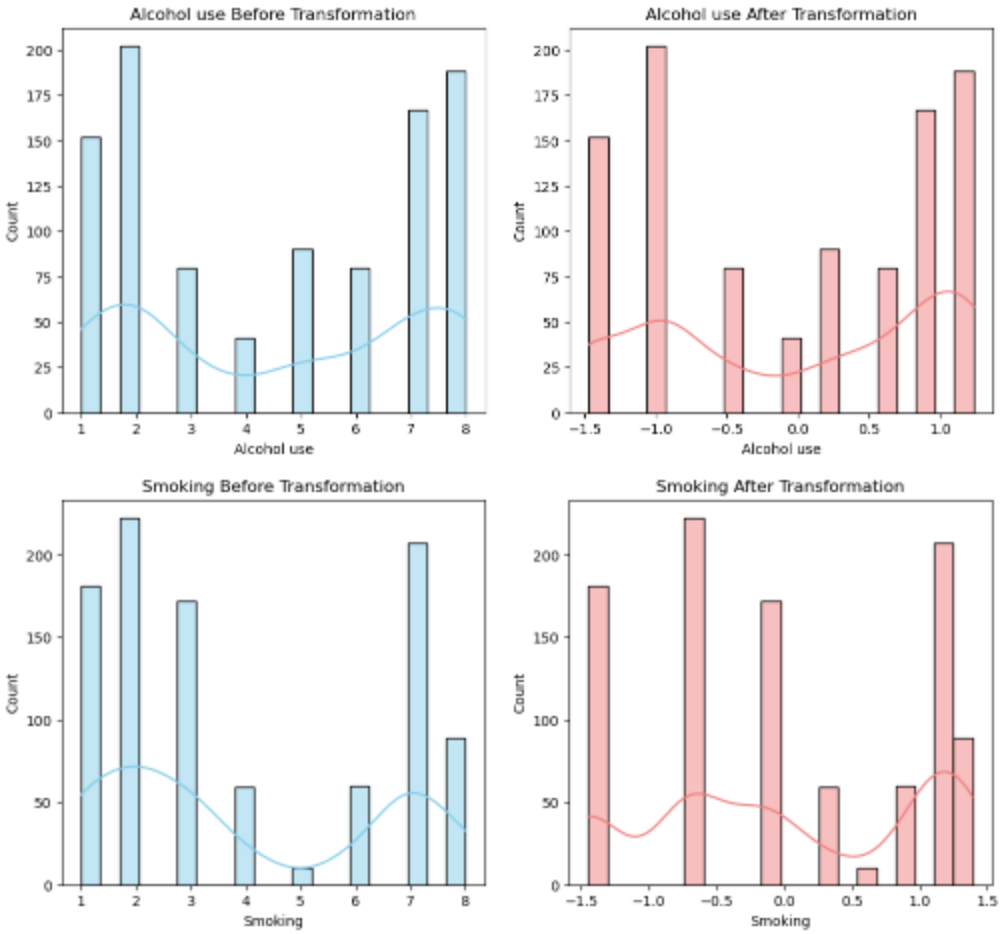
# EDA_Visualizations

## IT24101102 - Mohotti C.H. (Handling missing data)

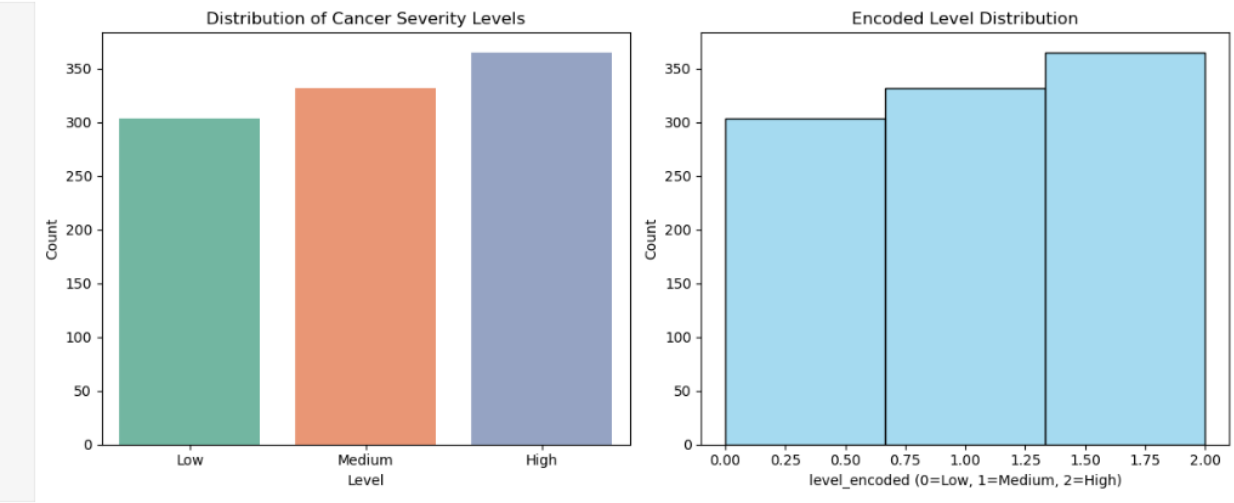
```
# Plot histograms for a few sample columns
sample_cols = ["Age", "Air Pollution", "Alcohol use", "Smoking"]

for col in sample_cols:
    plt.figure(figsize=(12,5))

    plt.subplot(1,2,1)
    sns.histplot(df[col], bins=20, kde=True, color="skyblue")
    plt.title(f"{col} Before Transformation")

    plt.subplot(1,2,2)
    sns.histplot(df_transformed[col], bins=20, kde=True, color="lightcoral")
    plt.title(f"{col} After Transformation")

    plt.show()
```





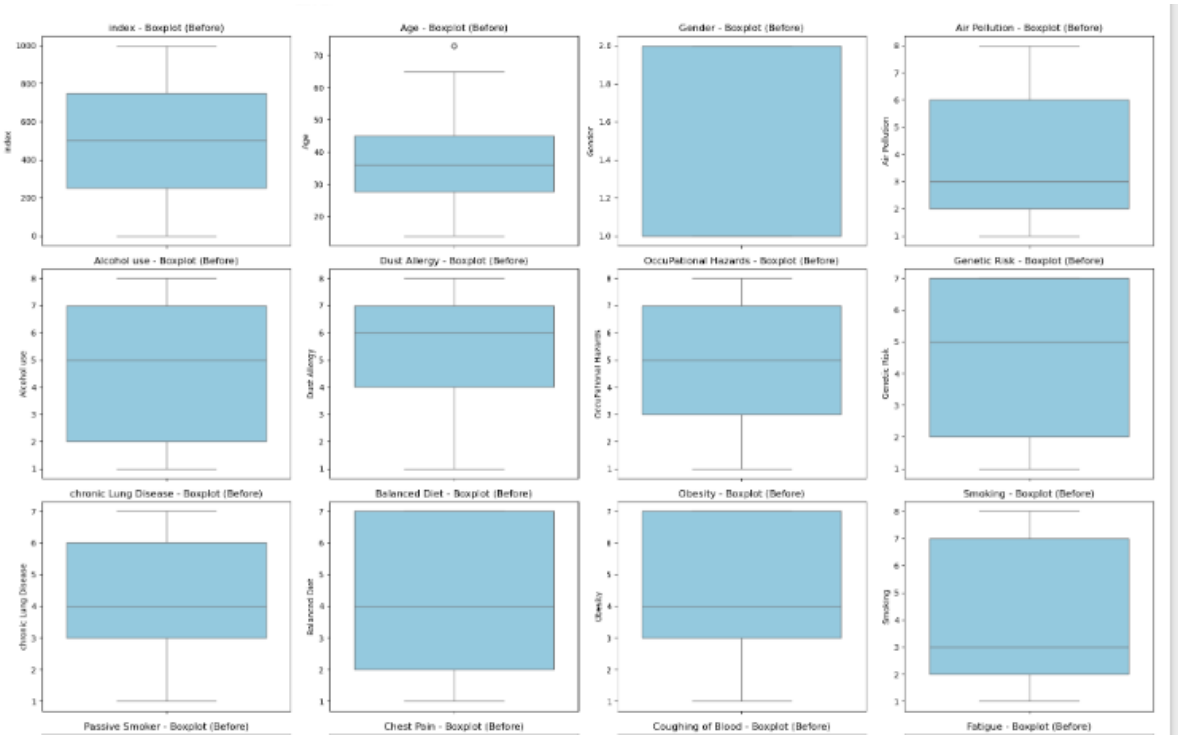

## IT24101206 - Karunarathne N.W. (Encoding categorical variables)

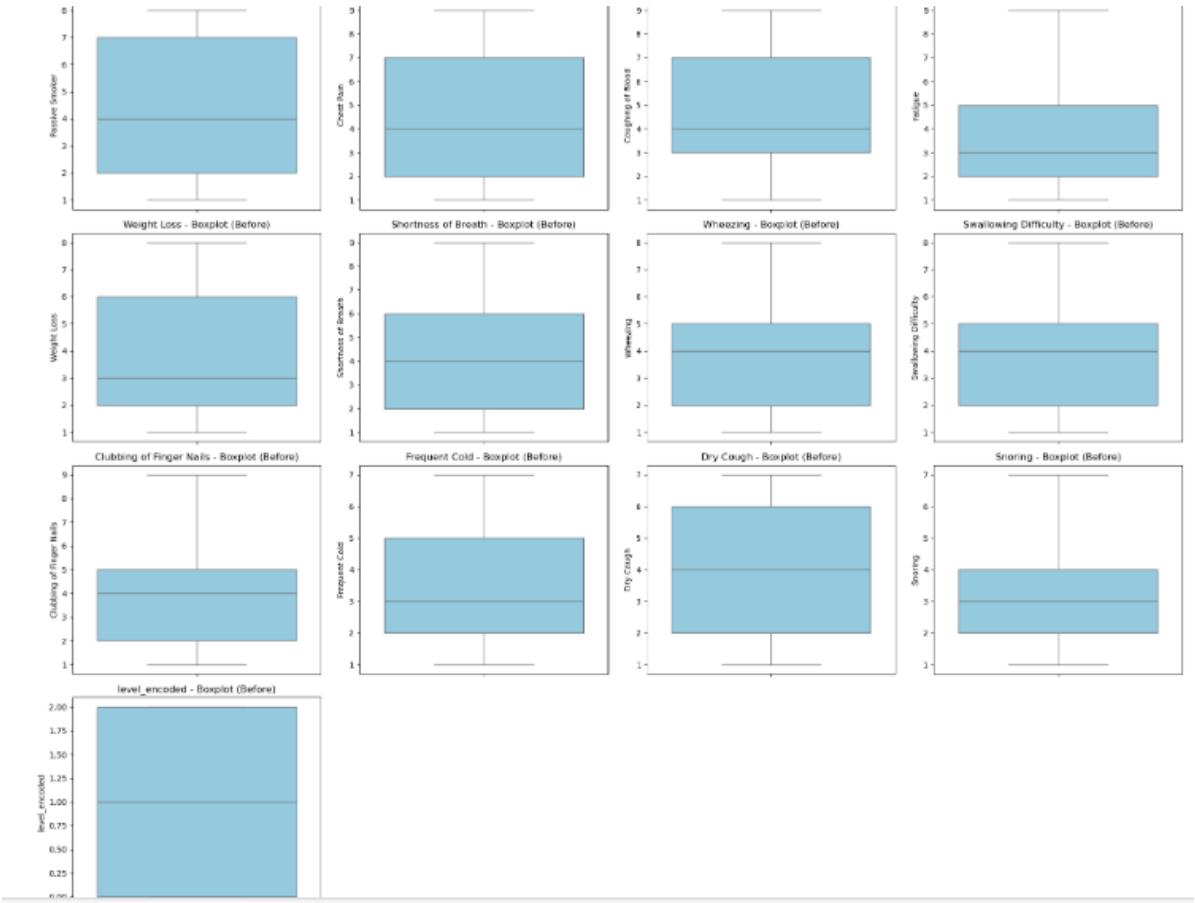
```
# Compare original Level vs encoded values
fig, axes = plt.subplots(1, 2, figsize=(12,5))

# Left: Categorical distribution (fixed for Seaborn 0.14+)
sns.countplot(x="Level", hue="Level", data=df,
              order=["Low", "Medium", "High"],
              palette="Set2", ax=axes[0], legend=False)
axes[0].set_title("Distribution of Cancer Severity Levels")
axes[0].set_xlabel("Level")
axes[0].set_ylabel("Count")

# Right: Encoded numeric distribution
sns.histplot(df["level_encoded"].dropna(), bins=3, kde=False, ax=axes[1], color="skyblue")
axes[1].set_title("Encoded Level Distribution")
axes[1].set_xlabel("level_encoded (0=Low, 1=Medium, 2=High)")
axes[1].set_ylabel("Count")

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



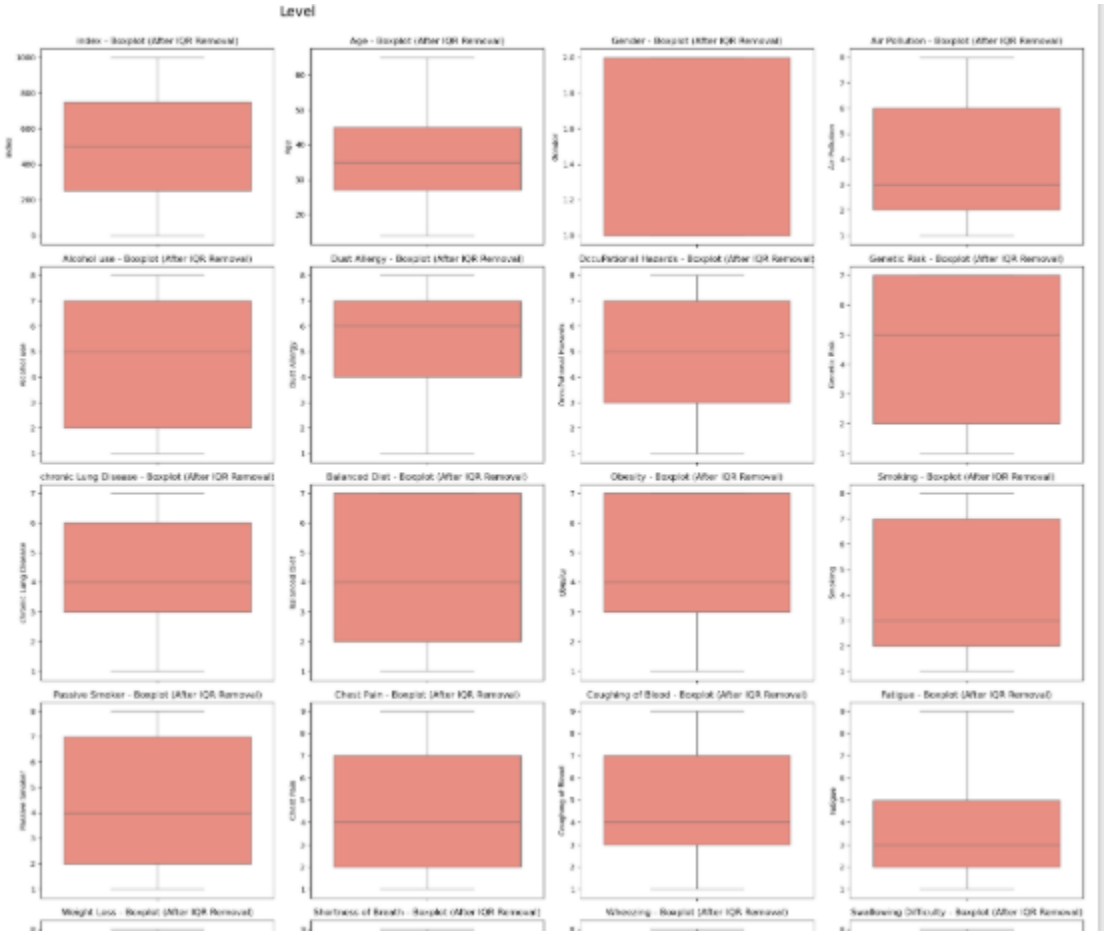

## IT24300015 - Da Silva H.V.N.N (Outlier removal)

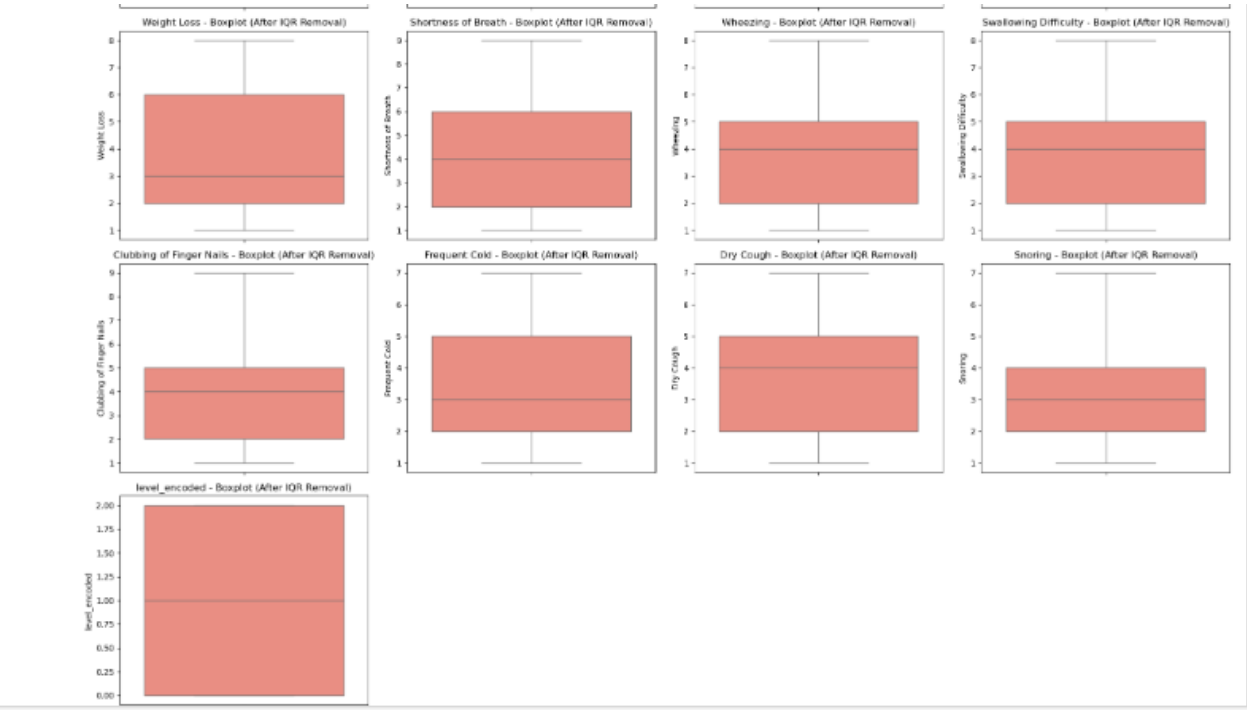
```
# Target variable distribution (Before)
plt.figure(figsize=(6,4))
sns.countplot(x="Level", data=df, hue="Level", palette="viridis", legend=False)
plt.title("Target Variable Distribution (Before)")
plt.show()

# Boxplots (Before Outlier Removal)
num_cols = df.select_dtypes(include=[np.number]).columns
n = len(num_cols)
rows = math.ceil(n / 4)   # dynamic rows based on number of numeric columns
cols = 4

plt.figure(figsize=(20, rows*4))
for i, col in enumerate(num_cols, 1):
    plt.subplot(rows, cols, i)
    sns.boxplot(y=df[col], color="skyblue")
    plt.title(f"{col} - Boxplot (Before)")

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


index - Boxplot (Before)
Age - Boxplot (Before)
Gender - Boxplot (Before)
Air Pollution - Boxplot (Before)
Alcohol use - Boxplot (Before)
Dust Allergy - Boxplot (Before)
OccuPational Hazards - Boxplot (Before)
Genetic Risk - Boxplot (Before)
chronic Lung Disease - Boxplot (Before)
Balanced Diet - Boxplot (Before)
Obesity - Boxplot (Before)
Smoking - Boxplot (Before)
Passive Smoker - Boxplot (Before)
Chest Pain - Boxplot (Before)
Coughing of Blood - Boxplot (Before)
Fatigue - Boxplot (Before)


Weight Loss - Boxplot (Before)
Shortness of Breath - Boxplot (Before)
Wheezing - Boxplot (Before)
Swallowing Difficulty - Boxplot (Before)
Clubbing of Finger Nails - Boxplot (Before)
Frequent Cold - Boxplot (Before)
Dry Cough - Boxplot (Before)
Snoring - Boxplot (Before)
level_encoded - Boxplot (Before)

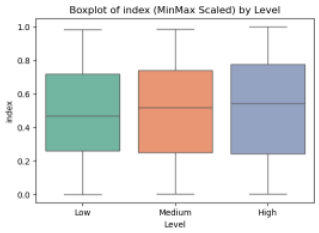
```
# EDA - After Outlier Removal

# Target variable distribution
plt.figure(figsize=(6,4))
sns.countplot(x="Level", data=df_iqr_removed, hue="Level", palette="magma", legend=False)
plt.title("Target Variable Distribution (After IQR Removal)")
plt.show()

# Boxplots (After IQR Removal)
num_cols = df_iqr_removed.select_dtypes(include=[np.number]).columns
n = len(num_cols)
rows = math.ceil(n / 4)
cols = 4
plt.figure(figsize=(20, rows*4))
for i, col in enumerate(num_cols, 1):
    plt.subplot(rows, cols, i)
    sns.boxplot(y=df_iqr_removed[col], color="salmon")
    plt.title(f"{col} - Boxplot (After IQR Removal)")

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

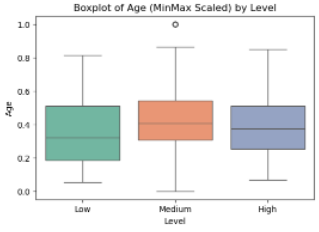


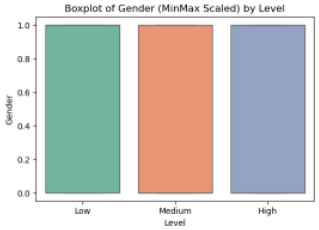


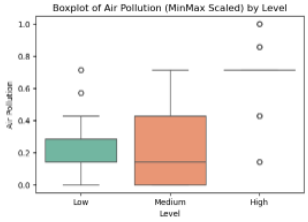

## IT24101093 - Chathurya K.G. Y (Normalization / Scaling (keeping Level as target)

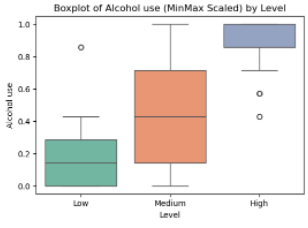
```
# 3. Boxplots after MinMax Scaling
for col in numeric_cols:
    plt.figure(figsize=(6,4))
    sns.boxplot(x="Level", y=col, data=df_minmax, hue="Level", palette="Set2", legend=False)
    plt.title(f"Boxplot of {col} (MinMax Scaled) by Level")
    plt.show()
```

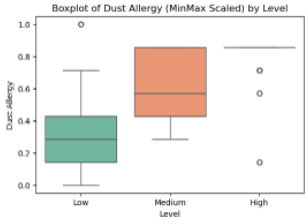


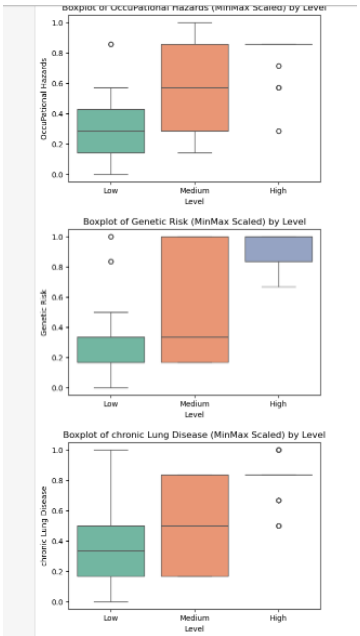



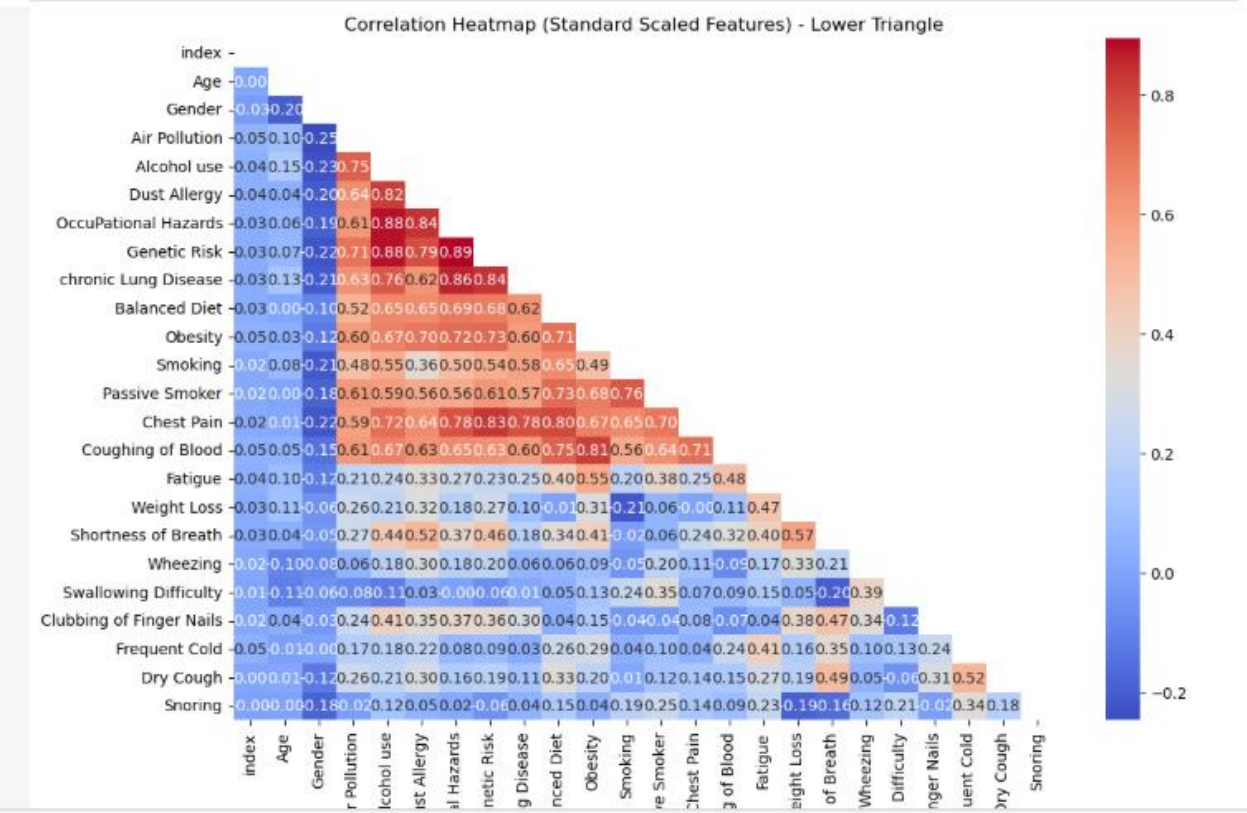



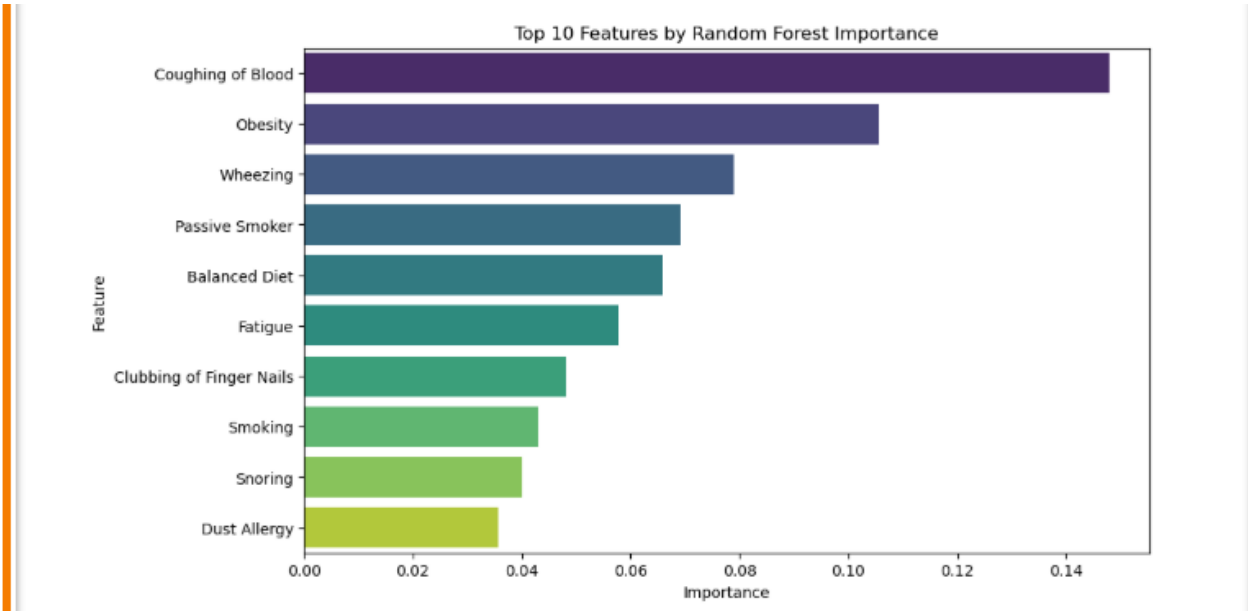



[87]:
```
# Select numeric columns only
num_cols = df.select_dtypes(include=['int64','float64']).columns.tolist()

# (Optional) remove target column if present
if "level_encoded" in num_cols:
    num_cols.remove("level_encoded")

# Mask the upper triangle
mask = np.triu(np.ones_like(df[num_cols].corr(), dtype=bool))

plt.figure(figsize=(12,8))
sns.heatmap(
    df[num_cols].corr(),
    cmap="coolwarm",
    annot=True,
    fmt=".2f",
    mask=mask,
    cbar=True
)
plt.title("Correlation Heatmap (Standard Scaled Features) - Lower Triangle")
plt.show()
```

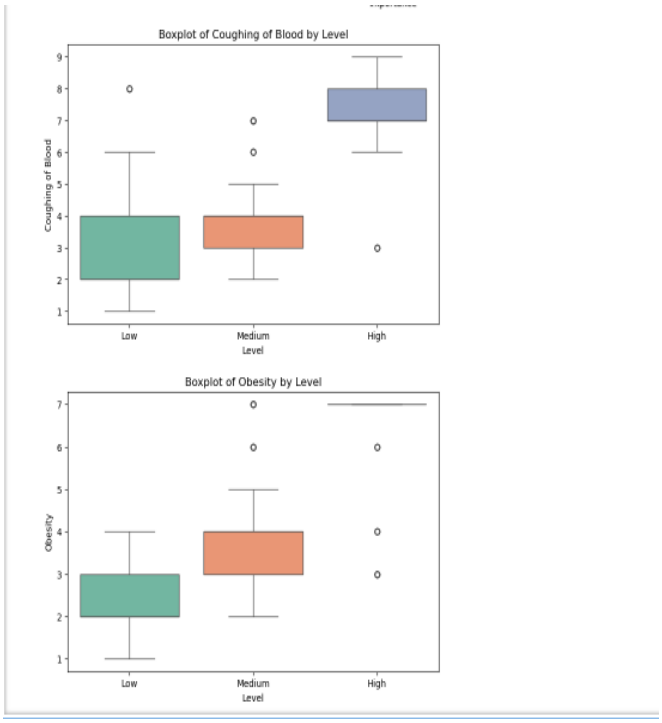


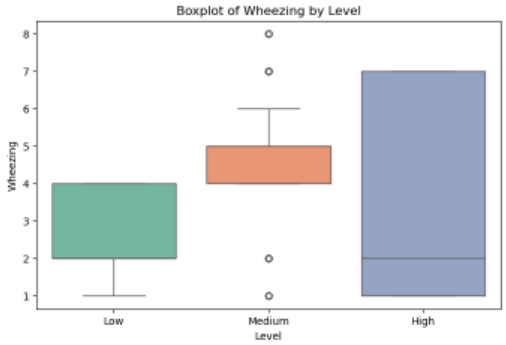
## IT24300019- Kumara M.P.H. S (Feature Engineering with Random Forest)

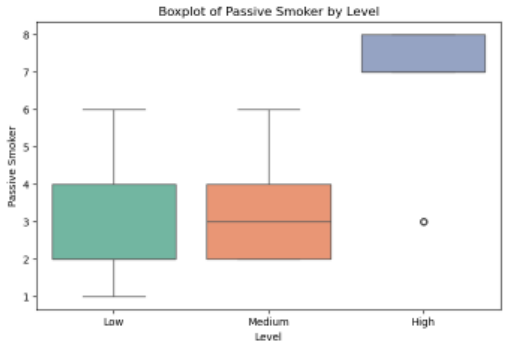
```
# 2. Barplot of Top 10 Features (fixed future warning)
plt.figure(figsize=(10,6))
sns.barplot(
    x="Importance",
    y="Feature",
    data=importance_df.head(10),
    hue="Feature",
    dodge=False,
    legend=False,
    palette="viridis"
)
plt.title("Top 10 Features by Random Forest Importance")
plt.show()
```



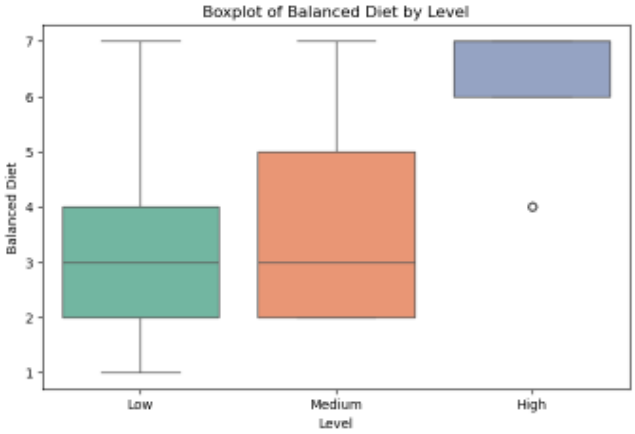

```
# Boxplots by Level (fixed future warning)
for col in top_features:
    plt.figure(figsize=(8, 5))
    sns.boxplot(
        x="Level",
        y=col,
        data=df_rf,
        hue="Level",
        dodge=False,
        legend=False,
        palette="Set2"
    )
    plt.title(f"Boxplot of {col} by Level")
    plt.show()
```


Boxplot of Coughing of Blood by Level
Coughing of Blood
Level
Low
Medium
High
Boxplot of Obesity by Level
Obesity
Level
Low
Medium
High


Boxplot of Wheezing by Level
Wheezing
Level
Low
Medium
High


Boxplot of Passive Smoker by Level
Passive Smoker
Level
Low
Medium
High

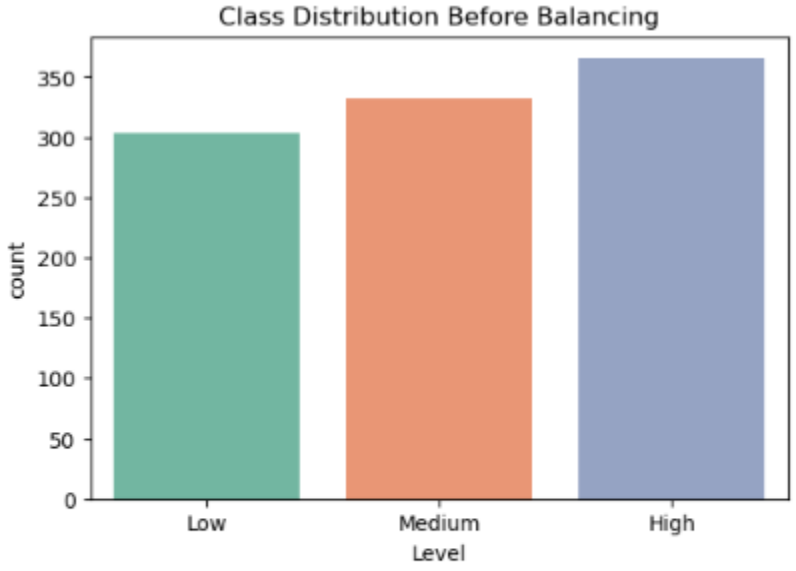
Level


Boxplot of Balanced Diet by Level
Balanced Diet
Level
Low
Medium
High

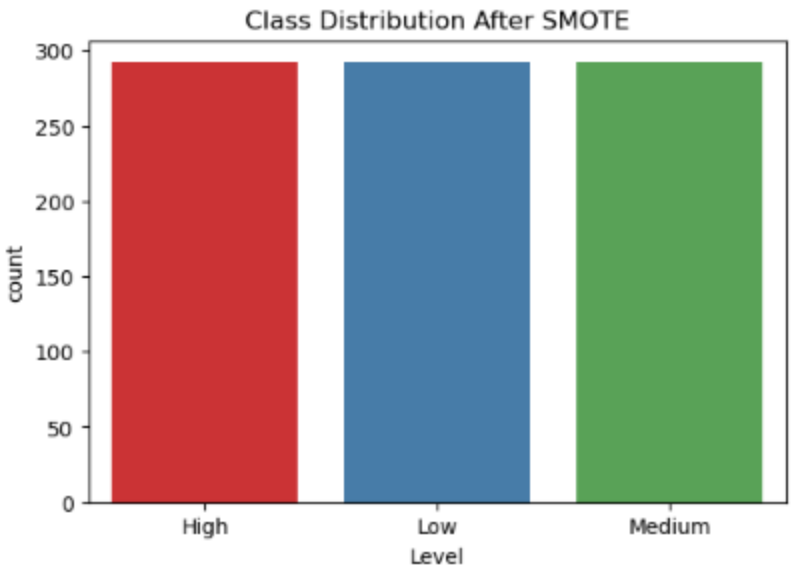
```
# Scatter plot with Legend
plt.figure(figsize=(8,6))
scatter = plt.scatter(pca_result[:, 0], pca_result[:, 1], c=y_numeric, cmap="viridis")

plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("PCA Dimensionality Reduction (Level Classes)")

# Create Legend with class names
handles, _ = scatter.legend_elements()
plt.legend(handles, le.classes_, title="Level")
plt.show()
```

**IT24101247-- Hettiarachchi D.G.P.M. (Class Imbalance)**

```
# Apply SMOTE only on training data
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Check new distribution after SMOTE
plt.figure(figsize=(6,4))
sns.countplot(x=y_train_resampled, hue=y_train_resampled, palette="Set1", legend=False)
plt.title("Class Distribution After SMOTE")
plt.show()
```




```
Class distribution:
 Level
High      36.5
Medium    33.2
Low       30.3
Name: proportion, dtype: float64
```




```
Resampled class distribution:
 Level
High      33.333333
Low       33.333333
Medium    33.333333
Name: proportion, dtype: float64
```