# Normality Test strategy

**Understanding the Importance of Normality Testing (and its Limitations)**

Before diving into the steps, it's crucial to understand *why* we often test for normality in data science:

- **Assumptions of Statistical Tests:** Many statistical tests (e.g., t-tests, ANOVA, linear regression) *assume* that the underlying data, or more specifically the *residuals* of a model, are normally distributed. Violating this assumption can lead to inaccurate p-values and confidence intervals, making your conclusions unreliable.
- **Feature Engineering:** Knowing the distribution of your features helps you choose appropriate data transformations (e.g., scaling, standardization) to improve model performance.
- **Model Selection:** Some models (like linear regression) are more sensitive to non-normality than others (like decision trees). Normality testing can inform your choice of model.

*Important Caveat:* Normality is *not always* required. Many machine learning algorithms (especially non-parametric ones like decision trees, random forests, and gradient boosting) are robust to non-normal data. Also, with large sample sizes, the Central Limit Theorem often comes into play, and the sampling distribution of the mean tends towards normality even if the underlying data isn't perfectly normal. So, while important, don't get *too* hung up on perfect normality.

**The Shapiro-Wilk Test: Interpreting the Results**

The Shapiro-Wilk test gives you a p-value. Here's how to interpret it:

- **p-value > Significance Level (usually 0.05):** You *fail to reject* the null hypothesis. This means there's not enough evidence to say the data is *not* normally distributed. For practical purposes, you can *assume* normality (but remember, it's not proof of normality).
- **p-value <= Significance Level (usually 0.05):** You *reject* the null hypothesis. This suggests that the data is likely *not* normally distributed.

**Decision Path: Your Next Steps**

Here's the decision tree, broken down by the results of your Shapiro-Wilk test, and including the rationale for each step:

**1. For Each Feature:**

- **Run Shapiro-Wilk Test:** Get the p-value.

**2. If the Feature IS Normally Distributed (p-value > 0.05):**

- **A. Consider Standardization (Z-scoring):**
- **Formula:** `z = (x - μ) / σ` (where x is the data point, μ is the mean, and σ is the standard deviation).
- **Why:** Standardization puts features on the same scale (mean of 0, standard deviation of 1). This is helpful for algorithms that are sensitive to feature magnitude (e.g., linear regression, k-Nearest Neighbors, Support Vector Machines). It doesn't *change* the distribution, but it makes it easier for these algorithms to work with.
- **When NOT to use:** If you're using tree-based models (Random Forest, Gradient Boosting), standardization usually isn't necessary, as these models are not sensitive to feature scaling.
- **B. Proceed with Modeling (if appropriate):** If the model you're using assumes normality (or benefits from it), and other assumptions are met, you can proceed.

**3. If the Feature IS NOT Normally Distributed (p-value <= 0.05):**

**A. Visual Inspection (Histogram, Q-Q Plot):**

- **Why:** The Shapiro-Wilk test is a statistical test, but a visual inspection is *crucial*. It helps you understand *how* the data deviates from normality (e.g., skewness, kurtosis, outliers).
- **Histogram:** Shows the frequency distribution of the data. Look for symmetry, unimodality (one peak), and bell-shape.
- **Q-Q Plot (Quantile-Quantile Plot):** Plots the quantiles of your data against the quantiles of a theoretical normal distribution. If the data is normal, the points should fall roughly along a straight diagonal line. Deviations from the line indicate non-normality.

**B. Identify the Type of Non-Normality:**

- **Skewness:** The data is lopsided (either to the left or right).
- **Kurtosis:** The data has heavier or lighter tails than a normal distribution (more or fewer extreme values).
- **Multimodality:** The data has multiple peaks.
- **Outliers:** Extreme values that significantly deviate from the rest of the data.

**C. Consider Data Transformations:**

- **Goal:** To make the data *more* normally distributed (or at least, less severely non-normal).
- **Common Transformations (and when to use them):**
- **Log Transformation:** `y = log(x)` (or `log(x+1)` if `x` can be 0).
- **Use for:** Right-skewed data (long tail to the right). This is *very* common in many datasets.
- **Square Root Transformation:** `y = sqrt(x)`.
- **Use for:** Moderately right-skewed data. Less powerful than log transformation.
- **Box-Cox Transformation:** A more general family of power transformations. It finds the *optimal* power transformation parameter (λ) to make the data as normal as possible. Most statistical software packages have a function for this (e.g., `boxcox` in SciPy).
- **Use for:** A variety of non-normal distributions. It's a good "go-to" if you're not sure which transformation to use.
- **Reciprocal Transformation:**`y = 1/x`.
- **Use For:**: Very, very skewed data
- **Yeo-Johnson Transformation:** Similar to Box-Cox, but it can handle zero and negative values.
- **Important Considerations:**
- **Interpretability:** Transformations can make your data less interpretable. Be sure you understand the implications of the transformation you choose.
- **Re-test for Normality:** After applying a transformation, *always* re-run the Shapiro-Wilk test and visual inspections to see if the transformation was effective.
- **Not Always Necessary:** As mentioned earlier, some models are robust to non-normality.

**D. Consider Outlier Handling (if applicable):**

- **Why:** Outliers can heavily influence normality tests and model performance.
- **Methods:**
- **Winsorizing:** Replace extreme values with less extreme values (e.g., replace values above the 99th percentile with the 99th percentile value).
- **Trimming:** Remove extreme values from the dataset. Use with caution, as you're losing data.
- **Robust Regression:** Use regression techniques that are less sensitive to outliers (e.g., robust regression).
- **Important:** Don't just blindly remove outliers. Investigate them! They might be genuine data points that represent important information, or they might be errors.

**E. Consider Non-Parametric Tests (if applicable):**

- **Why:** If your data remains significantly non-normal after transformation, and you need to perform statistical tests, consider using non-parametric tests. These tests *don't* assume normality.

**Examples:**

- **Mann-Whitney U test (instead of t-test):** Compares two independent groups.
- **Kruskal-Wallis test (instead of ANOVA):** Compares three or more independent groups.
- **Spearman's rank correlation (instead of Pearson correlation):** Measures the association between two variables.
- **F. Consider different model**
  - Why: if your data is not normal, consider using a model that does not assume normality.
  - Exemples:
    - Decision Trees
    - Random Forests
    - Gradient Boosting Machines (GBMs) like XGBoost, LightGBM, CatBoost
    - Support Vector Machines (SVMs)
    - k-Nearest Neighbors (k-NN)
    - Neural Networks

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
# Assume you have a DataFrame called 'df' and a
feature called 'feature_name'

# 1. Shapiro-Wilk Test
statistic, p_value =
stats.shapiro(df['feature_name'])
print(f"Shapiro-Wilk Test: Statistic={statistic:.3f},
p-value={p_value:.3f}")
if p_value > 0.05:
   print("Feature is likely normally distributed.")
   # 2A. Standardization (Optional)
   scaler = StandardScaler()
   df['feature_name_scaled'] =
scaler.fit_transform(df[['feature_name']])
```

```
    # Proceed with modeling

else:
    print("Feature is likely NOT normally
distributed.")

    # 3A. Visual Inspection
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.hist(df['feature_name'], bins=30)
    plt.title('Histogram')

    plt.subplot(1, 2, 2)
    sm.qqplot(df['feature_name'], line='s')  # 's' for
standardized line
    plt.title('Q-Q Plot')

    plt.show()

    # 3B. Identify the type of non-normality (based on
visual inspection)

    # 3C. Data Transformations (Example: Log
Transformation)
    if "right-skewed" in "your assessment":  # Replace
with your actual assessment
        df['feature_name_log'] =
np.log1p(df['feature_name'])  # Use log1p to handle
potential 0 values

        # Re-test for normality
        statistic, p_value =
stats.shapiro(df['feature_name_log'])
```

```
        print(f"Shapiro-Wilk Test (after log):
Statistic={statistic:.3f}, p-value={p_value:.3f}")

        # Re-visualize
        plt.figure(figsize=(12, 4))
        plt.subplot(1, 2, 1)
        plt.hist(df['feature_name_log'], bins=30)
        plt.title('Histogram (Log Transformed)')
        plt.subplot(1, 2, 2)
        sm.qqplot(df['feature_name_log'], line='s')
        plt.title('Q-Q Plot (Log Transformed)')
        plt.show()
    # 3.D Outlier handling
    # 3.E Consider Non-Parametric test
    # 3.F Consider different model

# Continue with your analysis, using the appropriate
transformed or original features.
```

**Key Takeaways:**

- Normality testing is a tool, not a rigid requirement. Understand the *assumptions* of your chosen methods.
- Visual inspection is *essential* alongside statistical tests.
- Data transformations can help, but they're not a magic bullet.
- Consider the robustness of your chosen model to non-normality.
- Document your decisions and transformations clearly.