Milestone 3

Power Analysis

Estee Lauder 1B

01 Method 1

Standard Power Analysis (T-Test)

Method 1: Standard Power Analysis (T-Test)

```
from statsmodels.stats.power import TTestIndPower

mde_percent = 0.01
mean_revenue = df['revenue (t)'].mean()
diff = mde_percent * mean_revenue

analysis = TTestIndPower()
effect_size = diff / df['revenue (t)'].std()
sample_size = analysis.solve_power(
    effect_size=effect_size,
    power=0.9,
    alpha=0.05,
    alternative="two-sided",
)
print("Required sample size per group (T-Test):", round(sample_size))
```

```
Required sample size per group (T-Test): 8801
```

Method 1: Standard Power Analysis (T-Test)

The power analysis using the t-test method shows that detecting a **minimum detectable effect** (MDE) of just **1%** of **mean revenue** with **90% power** at a **5% significance level** requires approximately **8,801 observations per group**, or about **17,600** in **total**.

This large sample size reflects the fact that very **small effects are statistically harder to detect**, since the difference between groups is small relative to the natural variation in revenue.

02 Method 2

MLRATE Power Analysis (Variance-reduced)

Method 2: MLRATE Power Analysis

```
# Step 0: Define outcome and features
# y is our target (revenue at time t) that we want to explain
y = df['revenue (t)'].values

# x are the predictors (features from time t-1 and loyalty status)
# these will help us explain some of the variance in revenue
X = df[['aov (t-1)',
        'days_since_last_purchase (t-1)',
        'tenure_in_days(t-1)',
        'loyalty_membership']].values
```

Method 2: MLRATE Power Analysis

```
# Step 1: Cross-fitting predictions
# we split the data into 2 folds so we can predict on data the model hasn't seen
kf = KFold(n_splits=2, shuffle=True, random_state=42)

# create empty array to store predictions
G = np.zeros_like(y)

# train on one half, predict on the other half
for train_idx, test_idx in kf.split(X):
    model = RandomForestRegressor(random_state=42)
    model.fit(X[train_idx], y[train_idx])        # train model
    G[test_idx] = model.predict(X[test_idx])     # save predictions
```

Method 2: MLRATE Power Analysis

```
# Step 2: Fit OLS with G
# add a constant (intercept) to predictions
G = sm.add_constant(G)

# regress actual revenue on predicted revenue
# residuals here = the part of revenue we still can't explain
ols_model = sm.OLS(y, G).fit(cov_type="HC0")
```

Method 2: MLRATE Power Analysis

```
# Step 3: Extract residuals & std
# get the unexplained part of revenue (residuals)
residuals = ols_model.resid

# calculate the standard deviation of residuals
resid_std = residuals.std(ddof=1)
```

Method 2: MLRATE Power Analysis

```
# Step 4: Compute new effect size (MLRATE)
# define minimum detectable effect (mde) as 1% of average revenue
mde_percent = 0.01
mean_revenue = df['revenue (t)'].mean()
diff = mde_percent * mean_revenue

# effect size = mde / std of residuals
effect_size_mlr = diff / resid_std
```

Method 2: MLRATE Power Analysis

```
# Step 5: Power analysis with new effect size
# calculate required sample size per group (90% power, 5% alpha, two-sided test)
analysis = TTestIndPower()
sample_size_mlr = analysis.solve_power(
    effect_size=effect_size_mlr,
    power=0.9,
    alpha=0.05,
    alternative="two-sided",
)

print("Required sample size per group (MLRATE):", round(sample_size_mlr))
```

```
Required sample size per group (MLRATE): 2778
```

Method 2: MLRATE Power Analysis

The power analysis using the MLRATE method shows that detecting a **minimum detectable effect** (MDE) of just **1%** of **mean revenue** with **90% power** at a **5% significance level** requires approximately **2,778 observations per group**, or about **5,556 in total.**

This reduced sample size reflects the fact that **variance reduction** through **covariate adjustment** makes it **easier to detect small effects**, since much of the natural variation in revenue is explained away by prior customer behavior.

03 Comparison

Method 1 vs Method 2

Method 1 vs Method 2: Comparison

Method 1: Standard Power Analysis (T-Test)

- Required sample size per group: **8,801** (≈17,600 total)
- Large sample size because **small effects are harder to detect** due to natural variance in revenue.

Method 2: MLRATE Power Analysis (Variance-Based)

- Required sample size per group: **2,778** (≈5,556 total, smaller amount than T-Test result)
- Smaller sample size because adjusting for covariates reduces unexplained variance, making **small effects easier to detect**.

MLRATE reduces the required sample size, demonstrating the benefit of variance reduction through covariate adjustment.

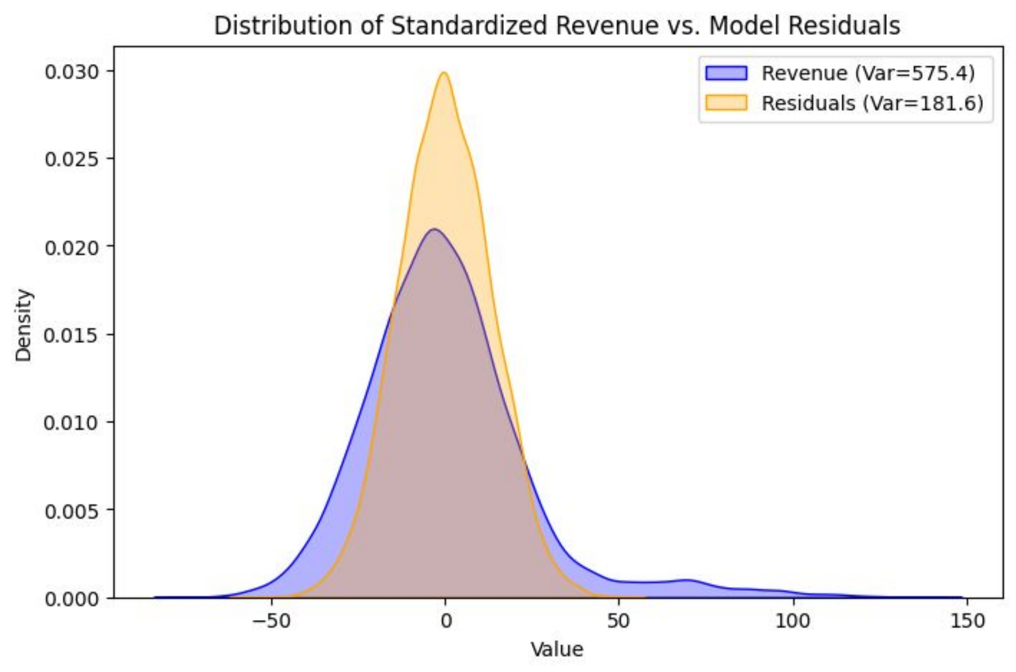
04 Plot

Residual Distribution Plot

Residual Distribution Plot Code

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

# Ensure numeric arrays
y = np.asarray(y, dtype=float)
residuals = np.asarray(residuals, dtype=float)

# Standardize (center both at 0)
revenue_std = y - np.mean(y)
residuals_std = residuals - np.mean(residuals)

# Compute variances
var_revenue = np.var(revenue_std, ddof=1)
var_residuals = np.var(residuals_std, ddof=1)

# Plot KDE curves
plt.figure(figsize=(8,5))
sns.kdeplot(revenue_std, color="blue", fill=True, alpha=0.3, label=f"Revenue (Var={var_revenue:.1f})")
sns.kdeplot(residuals_std, color="orange", fill=True, alpha=0.3, label=f"Residuals (Var={var_residuals:.1f})")

plt.title("Distribution of Standardized Revenue vs. Model Residuals")
plt.xlabel("Value")
plt.ylabel("Density")
plt.legend()
plt.show()
```

Residual Distribution Plot Result



Method 2 uses the MLRATE technique which significantly reduces variances. 181.6 / 575.4 = 0.316 and 1 = 0.316 is 0.684. So MLRATE reduces variance by 68.4%. MLRATE only needs 31.6% of the sample size from the t-test/ MLRATE cuts the sample size by 68%.

Thank you!

Notebook Link