Ketan_Attarde_24_27_06_Statistics_Codes

December 12, 2024

Ketan Attarde
M.Tech Data Science

24-27-06

Advanced Statistics Assignment Codes

1. Normal Distribution (Z)

[1]:
```python
import scipy.stats as stats

# Z-score to Probability (0 to Z)
def z_to_probability(z):
    return stats.norm.cdf(z) - 0.5

# Probability (0 to Z) to Z-score
def probability_to_z(prob):
    return stats.norm.ppf(prob + 0.5)

z = 1.96
prob = z_to_probability(z)
print(f"Probability (0 to {z}): {prob}")

p = 0.475
z_from_p = probability_to_z(p)
print(f"Z-score from probability {p}: {z_from_p}")
```

```
Probability (0 to 1.96): 0.4750021048517795
Z-score from probability 0.475: 1.959963984540054
```

2. T-Distribution Values

[2]:
```python
def t_value(prob, df):
    return stats.t.ppf(prob, df)

def probability_t(t_value, df):
    return stats.t.cdf(t_value, df)
```

```
prob = 0.975
df = 10  # Dof

t = t_value(prob, df)
print(f"T-value for probability {prob} and df {df}: {t}")

t_val = 2.228
prob = probability_t(t_val, df)
print(f"Probability for T-value {t_val} and df {df}: {prob}")
```

```
T-value for probability 0.975 and df 10: 2.2281388519649385
Probability for T-value 2.228 and df 10: 0.9749941140914443
```

3. Chi-Square Distribution

[3]:
```
def chi_square_value(prob, df):
    return stats.chi2.ppf(prob, df)

def probability_chi_square(chi_value, df):
    return stats.chi2.cdf(chi_value, df)


prob = 0.95
df = 5  # Dof
chi = chi_square_value(prob, df)
print(f"Chi-square value for probability {prob} and df {df}: {chi}")

chi_val = 11.07
prob = probability_chi_square(chi_val, df)
print(f"Probability for Chi-square value {chi_val} and df {df}: {prob}")
```

```
Chi-square value for probability 0.95 and df 5: 11.070497693516351
Probability for Chi-square value 11.07 and df 5: 0.9499903813775946
```

4. F- Distribution

[4]:
```
def f_value(prob, df1, df2):
    return stats.f.ppf(prob, df1, df2)

def probability_f(f_value, df1, df2):
    return stats.f.cdf(f_value, df1, df2)

prob = 0.95
df1, df2 = 5, 10  # Df
```

```python
f = f_value(prob, df1, df2)
print(f"F-value for probability {prob}, df1 {df1}, and df2 {df2}: {f}")

f_val = 3.33
prob = probability_f(f_val, df1, df2)
print(f"Probability for F-value {f_val}, df1 {df1}, and df2 {df2}: {prob}")
```

```
F-value for probability 0.95, df1 5, and df2 10: 3.3258345304130112
Probability for F-value 3.33, df1 5, and df2 10: 0.9501687242027786
```

5. Simple Linear Regression & ANOVA Table

[5]:
```python
import numpy as np
import pandas as pd

def regression_anova(x, y, alpha=0.05):
    n = len(x)
    x_mean = np.mean(x)
    y_mean = np.mean(y)

    # Calculate Beta values
    Sxy = np.sum((x - x_mean) * (y - y_mean))
    Sxx = np.sum((x - x_mean) ** 2)
    beta_1 = Sxy / Sxx
    beta_0 = y_mean

    # Predicted values and residuals
    y_pred = beta_0 + (x - x_mean) * beta_1
    residuals = y - y_pred

    #  ANOVA calculations
    SS_total = np.sum((y - y_mean) ** 2)
    SS_regression = np.sum((y_pred - y_mean) ** 2)
    SS_residual = np.sum((y - y_pred) ** 2)
    df_regression = 1
    df_residual = n - 2
    MS_regression = SS_regression / df_regression
    MS_residual = SS_residual / df_residual
    F_calculated = MS_regression / MS_residual

    # F-critical value from F-table
    F_table = stats.f.ppf(1 - alpha, df_regression, df_residual)

    # Hypothesis Test
    significance = "Significant" if F_calculated > F_table else "Not␣
↪Significant"
```

```python
    # ANOVA Table
    anova_table = pd.DataFrame({
        "Source": ["Regression", "Residual", "Total"],
        "SS": [SS_regression, SS_residual, SS_total],
        "df": [df_regression, df_residual, df_regression + df_residual],
        "MS": [MS_regression, MS_residual, None],
        "F": [F_calculated, None, None]
    })

    return anova_table, F_calculated, F_table, significance, beta_0, beta_1


x = np.array([1, 2, 3, 4, 5])
y = np.array([3, 5, 7, 9, 10])
anova_table, f_calc, f_tab, significance, beta_0, beta_1 = regression_anova(x,␣
 ↪y)


print("ANOVA Table:")
print(anova_table)
print(f"\nF-calculated: {f_calc}")
print(f"F-table (critical): {f_tab}")
print(f"Significance of using X: {significance}")
print(f"Regression Equation: Y = {beta_0:.2f} + (X - {np.mean(x):.2f}) *␣
 ↪{beta_1:.2f}")

# for y = theta_0 + theta_1 *x
theta_0 = beta_0 - (beta_1 * np.mean(x))
theta_1 = beta_1

print(f"Regression Equation: Y = {beta_0:.2f} + (X - {np.mean(x):.2f}) *␣
 ↪{beta_1:.2f}")
print(f" that is : Y = {theta_0:.2f} + (X * {theta_1:.2f})")
```

```
ANOVA Table:
       Source    SS  df         MS      F
0  Regression  32.4   1  32.400000  243.0
1    Residual   0.4   3   0.133333    NaN
2       Total  32.8   4        NaN    NaN

F-calculated: 242.99999999999974
F-table (critical): 10.127964486013928
Significance of using X: Significant
Regression Equation: Y = 6.80 + (X - 3.00) * 1.80
Regression Equation: Y = 6.80 + (X - 3.00) * 1.80
that is : Y = 1.40 + (X * 1.80)
```

[6]:
```python
import numpy as np
import pandas as pd

def multiple_regression_anova(X, Y, alpha=0.05):
    n, p = X.shape  # n x p

    # Add column of 1s to X
    X = np.hstack((np.ones((n, 1)), X))

    # Calculate coefficients (Beta) using the formula
    beta = np.linalg.inv(X.T @ X) @ (X.T @ Y)

    # Predicted values and residuals
    Y_pred = X @ beta
    residuals = Y - Y_pred

    #Sum of Squares
    SS_total = np.sum((Y - np.mean(Y)) ** 2)  # Total Sum of Squares (Syy)
    SS_regression = np.sum((Y_pred - np.mean(Y)) ** 2)  # Regression Sum of␣
↪Squares (SSR)
    SS_residual = np.sum((Y - Y_pred) ** 2)  # Residual Sum of Squares (SSE)

    #DOF
    df_regression = p
    df_residual = n - p - 1
    df_total = n - 1

    # Mean Squares
    MS_regression = SS_regression / df_regression
    MS_residual = SS_residual / df_residual

    #  F-Statistic
    F_calculated = MS_regression / MS_residual

    # Critical F-value
    F_table = stats.f.ppf(1 - alpha, df_regression, df_residual)

    # Hypothesis testing
    significance = "Significant" if F_calculated > F_table else "Not␣
↪Significant"

    # Create ANOVA Table
    anova_table = pd.DataFrame({
        "Source": ["Regression", "Residual", "Total"],
```

```python
        "SS": [SS_regression, SS_residual, SS_total],
        "df": [df_regression, df_residual, df_total],
        "MS": [MS_regression, MS_residual, None],
        "F": [F_calculated, None, None]
    })

    return anova_table, F_calculated, F_table, significance, beta


X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([3, 5, 7, 9, 10])

anova_table, f_calc, f_tab, significance, beta = multiple_regression_anova(X, Y)

# Hypothesis
print("Hypothesis Test:")
print("H0: All betas are 0 (X's are not significant)")
print("H1: At least one beta is non-zero (X's are significant)")

print("\nANOVA Table:")
print(anova_table)

print(f"\nF-calculated: {f_calc}")
print(f"F-table (critical): {f_tab}")
print(f"Significance: {significance}")

print("\nRegression Coefficients (Beta):")
for i, b in enumerate(beta):
    print(f"Beta_{i}: {b:.4f}")
```

```
Hypothesis Test:
H0: All betas are 0 (X's are not significant)
H1: At least one beta is non-zero (X's are significant)

ANOVA Table:
       Source    SS  df         MS      F
0  Regression  32.4   1  32.400000  243.0
1    Residual   0.4   3   0.133333    NaN
2       Total  32.8   4        NaN    NaN

F-calculated: 242.99999999999974
F-table (critical): 10.127964486013928
Significance: Significant

Regression Coefficients (Beta):
Beta_0: 1.4000
Beta_1: 1.8000
```