**Code:**

# Importing necessary libraries

import pandas as pd

import numpy as np

wcat = pd.read\_csv(":\\Users\\CSE-09\Downloads\\wc-at (1).csv")

C

# Exploratory data analysis:

# 1. Measures of central tendency

# 2. Measures of dispersion

# 3. Third moment business decision

# 4. Fourth moment business decision

# 5. Probability distributions of variables

# 6. Graphical representations (Histogram, Box plot, Dot plot, Stem & Leaf plot, Bar plot, etc.)

wcat.describe()

#Graphical Representation

import matplotlib.pyplot as plt # mostly used for visualization purposes

plt.bar(height = wcat.AT, x = np.arange(1, 110, 1))

plt.hist(wcat.AT) #histogram

plt.boxplot(wcat.AT) #boxplot

plt.bar(height = wcat.Waist, x = np.arange(1, 110, 1))

plt.hist(wcat.Waist) #histogram

plt.boxplot(wcat.Waist) #boxplot

# Scatter plot

plt.scatter(x = wcat['Waist'], y = wcat['AT'], color = 'blue')

# correlation

np.corrcoef(wcat.Waist, wcat.AT)

# Covariance

# NumPy does not have a function to calculate the covariance between two variables directly.

# Function for calculating a covariance matrix called cov()

# By default, the cov() function will calculate the unbiased or sample covariance between the provided random variables.

cov\_output = np.cov(wcat.Waist, wcat.AT)[0, 1]

cov\_output

# wcat.cov()

# Import library

import statsmodels.formula.api as smf

# Simple Linear Regression

model = smf.ols('AT ~ Waist', data = wcat).fit()

model.summary()

pred1 = model.predict(pd.DataFrame(wcat['Waist']))

# Regression Line

plt.scatter(wcat.Waist, wcat.AT)

plt.plot(wcat.Waist, pred1, "r")

plt.legend(['Predicted line', 'Observed data'])

plt.show()

# Error calculation

res1 = wcat.AT - pred1

res\_sqr1 = res1 \* res1

mse1 = np.mean(res\_sqr1)

rmse1 = np.sqrt(mse1)

rmse1

######### Model building on Transformed Data

# Log Transformation

# x = log(waist); y = at

plt.scatter(x = np.log(wcat['Waist']), y = wcat['AT'], color = 'brown')

np.corrcoef(np.log(wcat.Waist), wcat.AT) #correlation

model2 = smf.ols('AT ~ np.log(Waist)', data = wcat).fit()

model2.summary()

pred2 = model2.predict(pd.DataFrame(wcat['Waist']))

# Regression Line

plt.scatter(np.log(wcat.Waist), wcat.AT)

plt.plot(np.log(wcat.Waist), pred2, "r")

plt.legend(['Predicted line', 'Observed data'])

plt.show()

# Error calculation

res2 = wcat.AT - pred2

res\_sqr2 = res2 \* res2

mse2 = np.mean(res\_sqr2)

rmse2 = np.sqrt(mse2)

rmse2

#### Exponential transformation

# x = waist; y = log(at)

plt.scatter(x = wcat['Waist'], y = np.log(wcat['AT']), color = 'orange')

np.corrcoef(wcat.Waist, np.log(wcat.AT)) #correlation

model3 = smf.ols('np.log(AT) ~ Waist', data = wcat).fit()

model3.summary()

pred3 = model3.predict(pd.DataFrame(wcat['Waist']))

pred3\_at = np.exp(pred3)

pred3\_at

# Regression Line

plt.scatter(wcat.Waist, np.log(wcat.AT))

plt.plot(wcat.Waist, pred3, "r")

plt.legend(['Predicted line', 'Observed data'])

plt.show()

# Error calculation

res3 = wcat.AT - pred3\_at

res\_sqr3 = res3 \* res3

mse3 = np.mean(res\_sqr3)

rmse3 = np.sqrt(mse3)

rmse3

#### Polynomial transformation

# x = waist; x^2 = waist\*waist; y = log(at)

model4 = smf.ols('np.log(AT) ~ Waist + I(Waist\*Waist)', data = wcat).fit()

model4.summary()

pred4 = model4.predict(pd.DataFrame(wcat))

pred4\_at = np.exp(pred4)

pred4\_at

# Regression line

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 2)

X = wcat.iloc[:, 0:1].values

X\_poly = poly\_reg.fit\_transform(X)

# y = wcat.iloc[:, 1].values

plt.scatter(wcat.Waist, np.log(wcat.AT))

plt.plot(X, pred4, color = 'red')

plt.legend(['Predicted line', 'Observed data'])

plt.show()

# Error calculation

res4 = wcat.AT - pred4\_at

res\_sqr4 = res4 \* res4

mse4 = np.mean(res\_sqr4)

rmse4 = np.sqrt(mse4)

rmse4

# Choose the best model using RMSE

data = {"MODEL":pd.Series(["SLR", "Log model", "Exp model", "Poly model"]), "RMSE":pd.Series([rmse1, rmse2, rmse3, rmse4])}

table\_rmse = pd.DataFrame(wcat)

table rmse

###################

# The best model

from sklearn.model\_selection import train\_test\_split

train, test = train\_test\_split(wcat, test\_size = 0.2)

finalmodel = smf.ols('np.log(AT) ~ Waist + I(Waist\*Waist)', data = train).fit()

finalmodel.summary()

# Predict on test data

test\_pred = finalmodel.predict(pd.DataFrame(test))

pred\_test\_AT = np.exp(test\_pred)

pred\_test\_AT

# Model Evaluation on Test data

test\_res = test.AT - pred\_test\_AT

test\_sqrs = test\_res \* test\_res

test\_mse = np.mean(test\_sqrs)

test\_rmse = np.sqrt(test\_mse)

test\_rmse

# Prediction on train data

train\_pred = finalmodel.predict(pd.DataFrame(train))

pred\_train\_AT = np.exp(train\_pred)

pred\_train\_AT

# Model Evaluation on train data

train\_res = train.AT - pred\_train\_AT

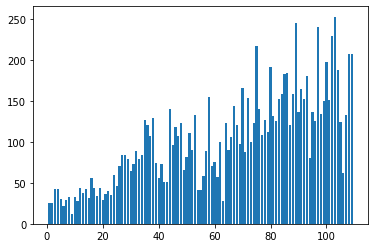
train\_sqrs = train\_res \* train\_res

train\_mse = np.mean(train\_sqrs)

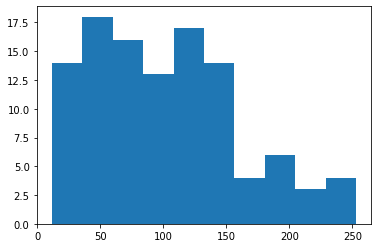
train\_rmse = np.sqrt(train\_mse)

train\_rmse

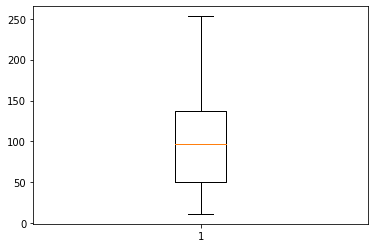
**Outputs:**

****

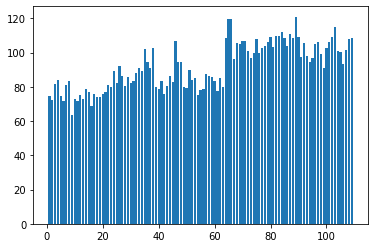
**Bar Graph for wcat. AT**



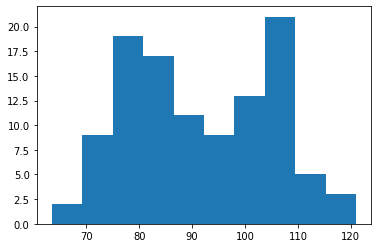
**Histogram for wcat. AT**



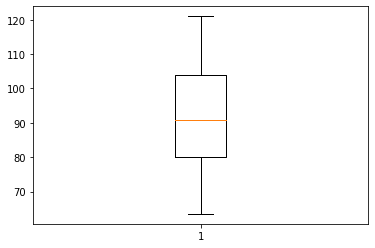
**Boxplot for wcat. AT**



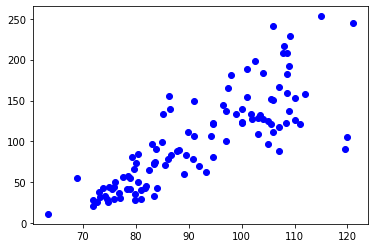
**Bar Graph for wcat. Waist**



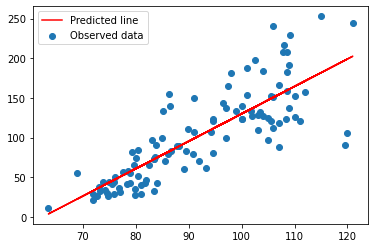
**Histogram for wcat. Waist**

****

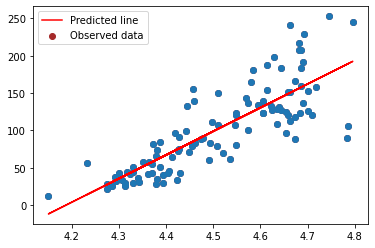
**Boxplot for wcat. Waist**



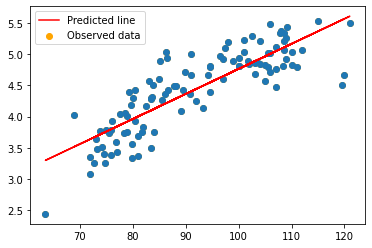
**Scatter plot show the split of relation in Waist and AT**



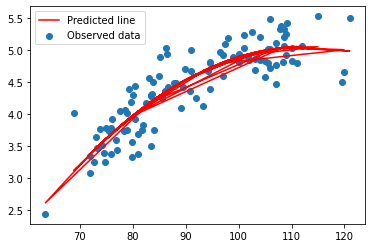
**Simple linear Model**

****

**Simple linear Model (Log)**



**Simple linear Model (Exponential)**



**Simple linear Model (Polynomial)**