Analysis of Concrete Compressive Strength Dataset

First A. Rida Syed, 2024540, Second B. Syeda Khadija Hassan, 2024626, and Third C. Maryam Khan, 2024281

Abstract--This project presents a comprehensive statistical analysis of concrete compressive strength data. Using a dataset of concrete samples with various component mixtures, we performed calculations of central tendency and dispersion through multiple methods. We established frequency distributions, created confidence and tolerance intervals, and tested hypotheses regarding the relationship between concrete components and strength. Our findings demonstrate the statistical properties of concrete strength and provide insights into the factors that influence concrete performance in construction applications

I. INTRODUCTION

The aim of this project is to analyze the compressive strength of concrete using statistical techniques implemented through Python. The study uses a dataset obtained from Kaggle, which contains data on the amounts of various components used in concrete mixes and the resulting compressive strength in megapascals. This dataset was selected due to its practical significance in civil and structural engineering applications and its richness in numerical data suitable for statistical analysis. We referenced Python's data analysis libraries such as pandas, numpy, matplotlib, and scipy to carry out the tasks. The dataset was cleaned to remove missing values and then statistically analyzed. Descriptive statistics such as mean and variance were computed using both built-in functions and frequency distribution-based approaches. Visualizations, including histograms and pie charts, were generated to aid understanding. Confidence intervals, tolerance intervals, and hypothesis tests were conducted to evaluate statistical properties and validate assumptions.

II. METHODOLOGY

A. Data Preparation

1) The dataset was loaded using pandas.read_csv().
2) Rows with missing values were dropped to ensure data quality.
3) The target variable, "Concrete compressive strength (MPa, megapascals)", was isolated for analysis.

B. Descriptive Statistics

1) Calculated mean and variance using built-in Python functions from the statistics module.
2) Estimated mean and variance using frequency distribution by binning the data into 100 bins using numpy.histogram().

C. Visualizations

1) Histogram of compressive strength distribution (Fig. 1).
2) Pie chart showing data distribution across 5 bins (Fig. 2).
3) Histogram annotated with built-in mean and standard deviation (Fig. 3).
4) Pie chart categorizing data into "Below Mean - SD", "Within ±SD", and "Above Mean + SD" (Fig. 4).
5) Histogram and pie chart based on frequency-derived mean and variance (Figs. 5 & 6).

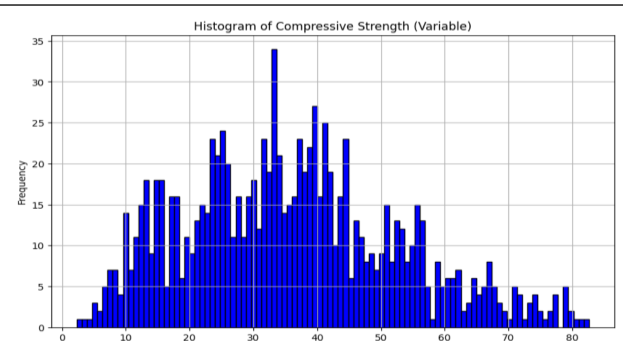


Fig. 1

Rida Syed, 2024540, Team lead, worked on finding the correct dataset, wrote codes on variance and average and pie charts, wrote half the report. Khadija Hassan, completed the report and found tables and graphs for the report, wrote code for frequency distribution, Maryam Khan, wrote code for confidence interval

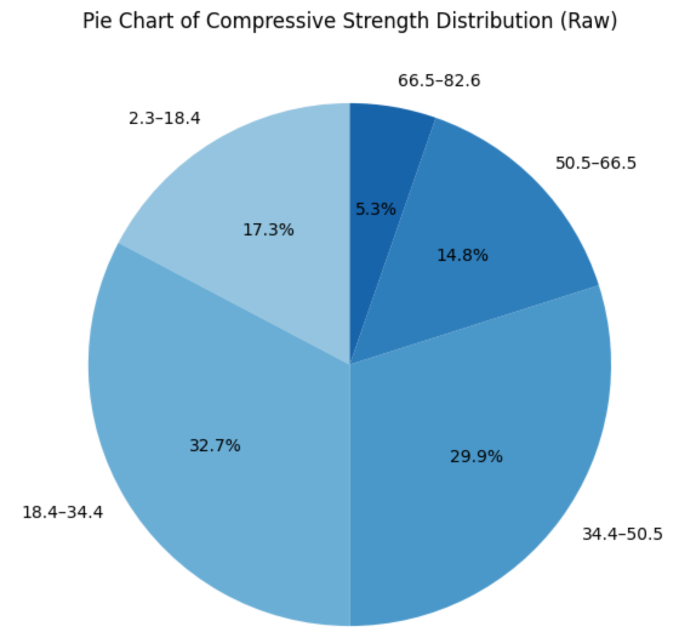


Fig. 2

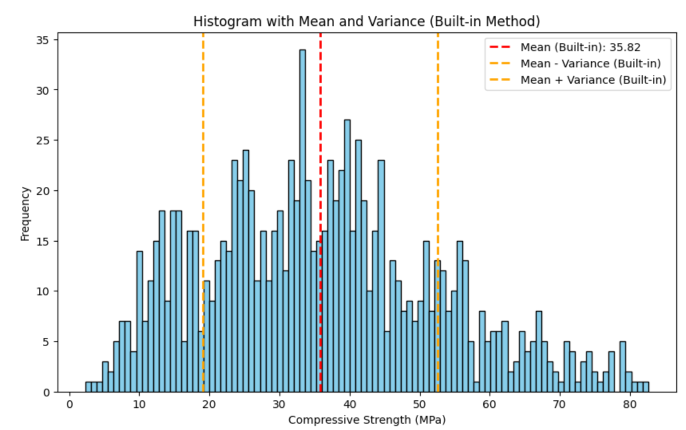


Fig. 3

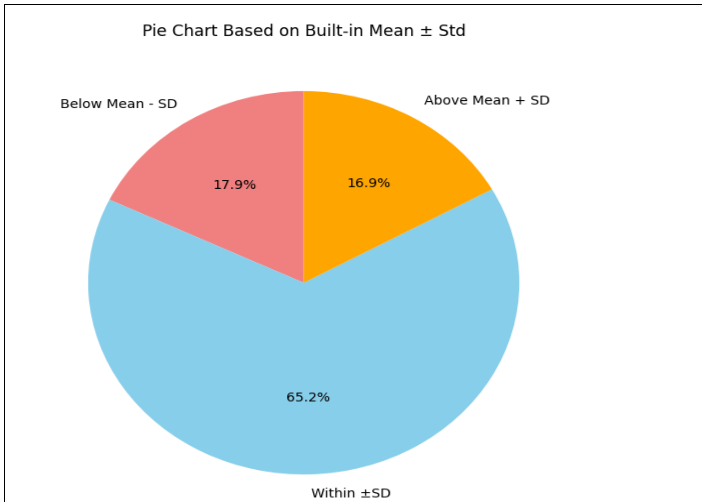
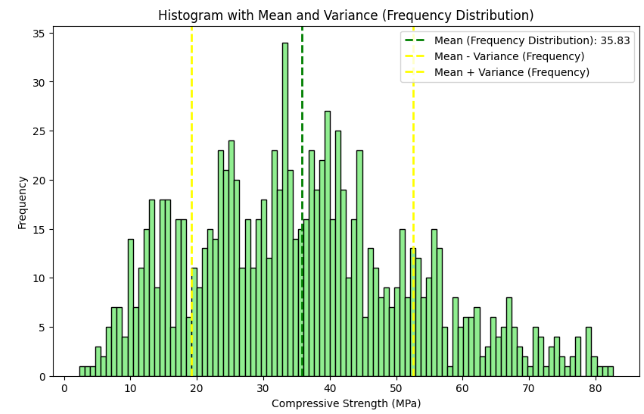


Fig. 4

Fig. 5



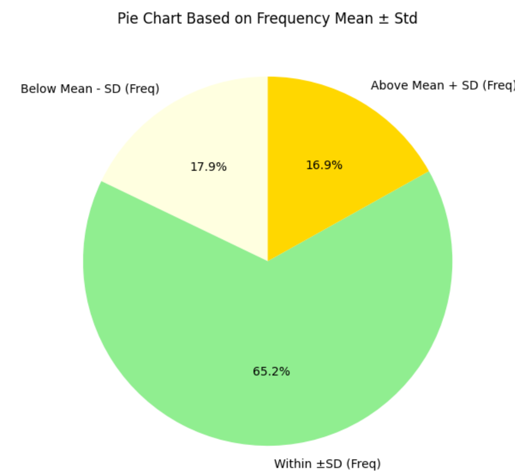


Fig. 6

D. Statistical Inference

1. Confidence Intervals
   - The dataset was split into 80% training and 20% validation.
   - 95% Confidence Interval for mean and variance was calculated using t and chi-square distributions, respectively.
2. Tolerance Interval
   - A 95% tolerance interval was calculated.
   - The remaining 20% of the data was used to validate this interval.
3. Hypothesis Testing
   - Null Hypothesis (H0): The mean compressive strength is less than or equal to 35 MPa.
   - Alternative Hypothesis (H1): The mean compressive strength is greater than 35 MPa.
   - One-tailed t-test was conducted using `scipy.stats.ttest_1samp()`.

III. RESULTS

A. Statistical Measures

We calculated the mean and variance using two methods: built-in functions and frequency distribution.

Using built-in functions:
$\mu = 35.817$ MPa
$\sigma^2 = 279.08 \mathrm{MPa}^2$

Using frequency distribution:
$\mu f = 35.832 \mathrm{MPa}$
$\sigma f^2 = 278.957 \mathrm{MPa}^2$

B. Figures:

- Fig. 1: Histogram of compressive strength
- Fig. 2: Pie chart with 5 bin segments
- Fig. 3: Histogram showing built-in mean ± std
- Fig. 4: Pie chart categorized by built-in mean ± std
- Fig. 5: Histogram showing frequency mean ± std
- Fig. 6: Pie chart categorized by frequency mean ± std

C. Distribution Analysis

The distribution analysis revealed that most samples (36.3%) fall in the 22.9-37.9 MPa range.

TABLE II
FREQUENCY DISTRIBUTION

Strength Range (MPa)	Percentage (%)
7.91 - 22.9	22.1
22.9 - 37.9	36.3
37.9 - 52.9	26.1
52.9 - 67.9	12.0
67.9 - 82.9	3.4

- Confidence Intervals (95%)

Let $\bar{x}$ be the sample mean, s the standard deviation, and n the sample size. The confidence interval is computed as:

(1) $\bar{x} \pm t\alpha/2, n-1\cdot(s/\sqrt{n})$

Result: Mean ∈ [35.36, 37.76]

- Variance Confidence Interval (95%)

Based on the chi-square distribution:

(2) $[(n-1)\cdot s^2/\chi^2 1-\alpha/2, (n-1)\cdot s^2/\chi^2\alpha/2]$

Result: Variance ∈ [280.28, 340.06]

- Tolerance Interval (95%)

The two-sided normal tolerance interval is given by:

(3) $\bar{x} \pm k\cdot s$

Result: Range = [2.14, 70.98], with 99.51% of values from validation set falling within this range.

- Hypothesis Testing

We tested:

H_0: $\mu \leq 35$ MPa (4)
H_1: $\mu > 35$ MPa (5)

T-statistic: 2.548
P-value: 0.01102
Reject the null hypothesis: Mean is significantly greater than 35 MPa.

IV. CONCLUSION

The analysis of the concrete compressive strength dataset highlights the value of statistical methods in engineering applications. By applying both built-in and frequency-distribution techniques, we were able to uncover consistent insights into the distribution of strength values. The use of confidence intervals and hypothesis testing provided strong evidence to support statistical assumptions about the dataset. These findings can support quality control in concrete production and guide materials engineering decisions. This work also sets a foundation for future enhancements such as predictive modeling or incorporating multivariate relationships among concrete ingredients and their effects on strength.

APPENDIX:

```
from google.colab import files
uploaded = files.upload()
#ES PROJECT- CONCRETE DATA
#2024540
#2024626
#2024281
import pandas as pd
import statistics as st
import matplotlib.pyplot as mb
import numpy as np

# Load the dataset
data = pd.read_csv("Concrete_Data.xls.csv")

# Clean the data by dropping all nan values in rows
data_cleaned = data.dropna(axis=0)

# Print the cleaned data
print("Cleaned Data:")
print(data_cleaned)

# Extract the target column
strength = data_cleaned['Concrete compressive
strength(MPa, megapascals) ']

# Built-in mean and variance
mean_builtin = st.mean(strength)
variance_builtin = st.variance(strength)
print(f"Mean (built-in): {mean_builtin}")
print(f"Variance (built-in): {variance_builtin}")

# Frequency distribution-based mean and variance
counts, bin_edges = np.histogram(strength,
bins=100)
mid_points = (bin_edges[1:] + bin_edges[:-1]) / 2

# Calculate mean from frequency distribution
mean_freq = np.sum(counts * mid_points) /
np.sum(counts)
```

```
# Calculate variance from frequency
distribution
variance_freq = np.sum(counts * (mid_points -
mean_freq)**2) / np.sum(counts)

print(f"Mean from frequency distribution:
{mean_freq}")
print(f"Variance from frequency distribution:
{variance_freq}")

# Plot 1: Histogram of the Variable
(Compressive Strength)
mb.figure(figsize=(10, 6))
mb.hist(strength, bins=100, color='blue',
edgecolor='black')
mb.title("Histogram of Compressive Strength
(Variable)")
mb.xlabel("Compressive Strength (MPa)")
mb.ylabel("Frequency")
mb.grid(True)
mb.show()

# Pie Chart 1: Distribution from raw
histogram (same bins)
counts1, bins1 = np.histogram(strength,
bins=5)  # Fewer bins for clearer pie chart
labels1 = [f"{round(bins1[i], 1)}-
{round(bins1[i+1], 1)}" for i in
range(len(bins1)-1)]

mb.figure(figsize=(7, 7))
mb.pie(counts1, labels=labels1,
autopct='%1.1f%%', startangle=90,
colors=mb.cm.Blues(np.linspace(0.4, 0.8,
len(counts1))))
mb.title("Pie Chart of Compressive Strength
Distribution (Raw)")
mb.show()


# Plot 2: Mean and Variance (Built-in Method)
mb.figure(figsize=(10, 6))
mb.hist(strength, bins=100, color='skyblue',
edgecolor='black')
mb.axvline(mean_builtin, color='red',
linestyle='dashed', linewidth=2, label=f'Mean
(Built-in): {mean_builtin:.2f}')
mb.axvline(mean_builtin -
np.sqrt(variance_builtin), color='orange',
linestyle='dashed', linewidth=2, label=f'Mean
- Variance (Built-in)')
mb.axvline(mean_builtin +
np.sqrt(variance_builtin), color='orange',
linestyle='dashed', linewidth=2, label=f'Mean
+ Variance (Built-in)')
mb.title("Histogram with Mean and Variance
(Built-in Method)")
mb.xlabel("Compressive Strength (MPa)")
mb.ylabel("Frequency")
mb.legend()
mb.show()

# Pie Chart 2: Categorize based on mean ± std
from built-in method
low = (strength < mean_builtin -
np.sqrt(variance_builtin)).sum()
mid = ((strength >= mean_builtin -
np.sqrt(variance_builtin)) & (strength <=
mean_builtin +
np.sqrt(variance_builtin))).sum()
high = (strength > mean_builtin +
np.sqrt(variance_builtin)).sum()

mb.figure(figsize=(7, 7))
mb.pie([low, mid, high],
       labels=["Below Mean - SD", "Within
±SD", "Above Mean + SD"],
       autopct='%1.1f%%',
       colors=['lightcoral', 'skyblue',
'orange'],
       startangle=90)
mb.title("Pie Chart Based on Built-in Mean ±
Std")
```

```
# Plot 3: Frequency-based Mean and Variance
mb.figure(figsize=(10, 6))
mb.hist(strength, bins=100, color='lightgreen',
edgecolor='black')
mb.axvline(mean_freq, color='green',
linestyle='dashed', linewidth=2, label=f'Mean
(Frequency Distribution): {mean_freq:.2f}')
mb.axvline(mean_freq - np.sqrt(variance_freq),
color='yellow', linestyle='dashed', linewidth=2,
label=f'Mean - Variance (Frequency)')
mb.axvline(mean_freq + np.sqrt(variance_freq),
color='yellow', linestyle='dashed', linewidth=2,
label=f'Mean + Variance (Frequency)')
mb.title("Histogram with Mean and Variance
(Frequency Distribution)")
mb.xlabel("Compressive Strength (MPa)")
mb.ylabel("Frequency")
mb.legend()
mb.show()

# Pie Chart 3: Based on frequency mean ± std
low_f = (strength < mean_freq -
np.sqrt(variance_freq)).sum()
mid_f = ((strength >= mean_freq -
np.sqrt(variance_freq)) & (strength <= mean_freq +
np.sqrt(variance_freq))).sum()
high_f = (strength > mean_freq +
np.sqrt(variance_freq)).sum()

mb.figure(figsize=(7, 7))
mb.pie([low_f, mid_f, high_f],
       labels=["Below Mean - SD (Freq)", "Within ±SD
(Freq)", "Above Mean + SD (Freq)"],
       autopct='%1.1f%%',
       colors=['lightyellow', 'lightgreen', 'gold'],
       startangle=90)
mb.title("Pie Chart Based on Frequency Mean ± Std")
mb.show()


from scipy import stats

# ------------------------------
# Step 1: Split the data (80% for analysis, 20% for
validation)
split_index = int(0.8 * len(strength))
strength_80 = strength.iloc[:split_index]
strength_20 = strength.iloc[split_index:]

# Step 2: Calculate sample statistics for 80%
n = len(strength_80)
sample_mean = st.mean(strength_80)
sample_std = st.stdev(strength_80)

# 95% Confidence Interval for the Mean (t-
distribution)
conf_level = 0.95
alpha = 1 - conf_level
t_crit = stats.t.ppf(1 - alpha/2, df=n-1)
margin_error = t_crit * (sample_std / np.sqrt(n))
ci_mean_lower = sample_mean - margin_error
ci_mean_upper = sample_mean + margin_error

print(f"\n95% Confidence Interval for Mean:
[{ci_mean_lower:.2f}, {ci_mean_upper:.2f}]")

# 95% Confidence Interval for Variance (chi-square
distribution)
sample_var = st.variance(strength_80)
chi2_lower = stats.chi2.ppf(alpha / 2, df=n - 1)
chi2_upper = stats.chi2.ppf(1 - alpha / 2, df=n - 1)
ci_var_lower = (n - 1) * sample_var / chi2_upper
ci_var_upper = (n - 1) * sample_var / chi2_lower

print(f"95% Confidence Interval for Variance:
[{ci_var_lower:.2f}, {ci_var_upper:.2f}]")
# Step 3: 95% Tolerance Interval (k-factor from
normal dist)
k_factor = stats.norm.ppf(1 - alpha / 2) * np.sqrt(1
+ 1/n)
tol_lower = sample_mean - k_factor * sample_std
tol_upper = sample_mean + k_factor * sample_std
```

```
print(f"95% Tolerance Interval:
[{tol_lower:.2f}, {tol_upper:.2f}]")

# Step 4: Validation using the remaining 20%
data
within_interval = ((strength_20 >= tol_lower)
& (strength_20 <= tol_upper)).sum()
total_20 = len(strength_20)
percentage_within = (within_interval /
total_20) * 100

print(f"Validation: {within_interval} out of
{total_20} ({percentage_within:.2f}%) values
lie within the 95% tolerance interval.")

# ------------------------------
# Step 5: Hypothesis Testing
# Hypothesis: H0: mean <= 35, H1: mean > 35
(One-tailed t-test)
hypothesized_mean = 35
t_statistic, p_value =
stats.ttest_1samp(strength_80,
popmean=hypothesized_mean)

print("\nHypothesis Test:")
print(f"T-statistic: {t_statistic:.3f}")
```

REFERENCES

[1] Kaggle, "Concrete Compressive Strength Data Set," https://www.kaggle.com/datasets/uciml/concrete-compressive-strength

[2] G. Van Rossum and F. L. Drake, "Python 3 Reference Manual," Python Software Foundation, 2009.

[3] W. McKinney, "Data Structures for Statistical Computing in Python," in Proc. 9th Python in Science Conf., Austin, TX, USA, 2010, pp. 51–56.

[4] J. D. Hunter, "Matplotlib: A 2D graphics environment," Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, May/Jun. 2007, doi: 10.1109/MCSE.2007.55.

[5] P. Virtanen et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," Nat. Methods, vol. 17, pp. 261–272, Mar. 2020, doi: 10.1038/s41592-019-0686-2.