**Batch: A1 Roll No.: 16010121045**

**Experiment / Assignment / Tutorial No. 7**

|  |
| --- |
| **Title: Implementation of input modeling steps for simulation** |

**Objective:** To understand and implement the process of developing input models for simulation by following the four essential steps: data collection, identifying probability distributions, choosing parameters for the distributions, and evaluating the goodness of fit. The experiment aims to highlight the significance of high-quality input data in producing reliable simulation outputs.

**Expected Outcome of Experiment:**

CO4: Analyze the systems for input modeling and validation.

CO5: Estimate the different parameters of absolute and relative performance of different simulation systems.

**Books/ Journals/ Websites referred:**

1. “Discrete-Event System Simulation” by Jerry Banks, John S. Carson II, Barry L. Nelson, David M. Nicol.
2. SimPy Documentation: https://simpy.readthedocs.io/en/latest/
3. SciPy Documentation: https://docs.scipy.org/doc/scipy/

**Background:**

(Explain in brief steps for input modeling development in simulation.)

**Problem Statement:**

Perform the following steps of Input Model Development:

1. **Collect Data from the Real System:**

* Identify the key input processes in the system (e.g., arrival times, service times).
* Collect data through direct observation, historical records, or sensors.
* Ensure data is recorded accurately and is sufficient in quantity to perform statistical analysis.

1. **Identify a Probability Distribution to Represent the Input Process:**

* Plot the collected data to visually inspect its distribution (e.g., using histograms).
* Use statistical software or tools to fit different probability distributions to the data.
* Compare the fit of different distributions using statistical measures (e.g., likelihood, AIC).

1. **Choose Parameters for the Distribution:**

* Use statistical software or tools to estimate the parameters of the chosen distribution.
* Ensure that the estimated parameters make sense in the context of the real system.

1. **Evaluate the Chosen Distribution and Parameters for Goodness of Fit:**

* Perform goodness-of-fit tests such as the Kolmogorov-Smirnov test, Chi-square test.
* Compare the observed data with the expected values from the chosen distribution.

**Implementation Steps with Screen shots:**

*import* pandas *as* pd

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*from* scipy *import* stats

df = pd.read\_csv('tsla\_2014\_2023.csv')

*# 2. Plot the volume column*

plt.hist(df['volume'], *bins*=100, *density*=True, *alpha*=0.7, *color*='g')

plt.title("Histogram of Tesla Stock volumes")

plt.xlabel("volume")

plt.ylabel("Density")

plt.show()

*# 3. Fit Probability Distributions: Fit Normal and Exponential distributions to the volume data*

fit\_expon = stats.expon.fit(df['volume'])

fit\_lognorm = stats.lognorm.fit(df['volume'])

*# Plot the fitted distributions*

x = np.linspace(min(df['volume']), max(df['volume']), 100)

pdf\_expon = stats.expon.pdf(x, \*fit\_expon)

pdf\_lognorm = stats.lognorm.pdf(x, \*fit\_lognorm)

plt.hist(df['volume'], *bins*=100, *density*=True, *alpha*=0.6, *color*='gray')

plt.plot(x, pdf\_expon, 'r-', *label*="Exponential Fit")

plt.plot(x, pdf\_lognorm, 'b-', *label*="Log Normal Fit")

plt.title("Fitting Distributions to volumes")

plt.xlabel("volume")

plt.ylabel("Density")

plt.legend()

plt.show()

*# 4. Extract Parameters for the Fitted Distributions*

print(f"Exponential Fit Parameters (loc, scale): {fit\_expon}")

print(f"Log-Normal Fit Parameters (shape, loc, scale): {fit\_lognorm}")

*# 5. Goodness-of-Fit Testing using Kolmogorov-Smirnov (KS) test*

ks\_expon = stats.kstest(df['volume'], 'expon', *args*=fit\_expon)

print(f"Kolmogorov-Smirnov test for Exponential: {ks\_expon}")

ks\_lognorm = stats.kstest(df['volume'], 'lognorm', *args*=fit\_lognorm)

print(f"Kolmogorov-Smirnov test for Log Normal: {ks\_lognorm}")

*# 6. Chi-square goodness-of-fit test*

*# Observed frequencies (histogram data)*

observed\_freq, bins = np.histogram(df['volume'], *bins*=100)

*# Expected frequencies (Exponential and Log-Normal)*

expected\_freq\_expon = stats.expon.cdf(bins[1:], \*fit\_expon) - stats.expon.cdf(bins[:-1], \*fit\_expon)

expected\_freq\_lognorm = stats.lognorm.cdf(bins[1:], \*fit\_lognorm) - stats.lognorm.cdf(bins[:-1], \*fit\_lognorm)

*# Normalize expected frequencies to match the observed sum*

expected\_freq\_expon \*= len(df['volume'])

expected\_freq\_lognorm \*= len(df['volume'])

*# Scale expected frequencies to match the total observed sum*

expected\_freq\_expon \*= observed\_freq.sum() / expected\_freq\_expon.sum()

expected\_freq\_lognorm \*= observed\_freq.sum() / expected\_freq\_lognorm.sum()

*# Perform the chi-square test*

chi\_square\_expon = stats.chisquare(*f\_obs*=observed\_freq, *f\_exp*=expected\_freq\_expon)

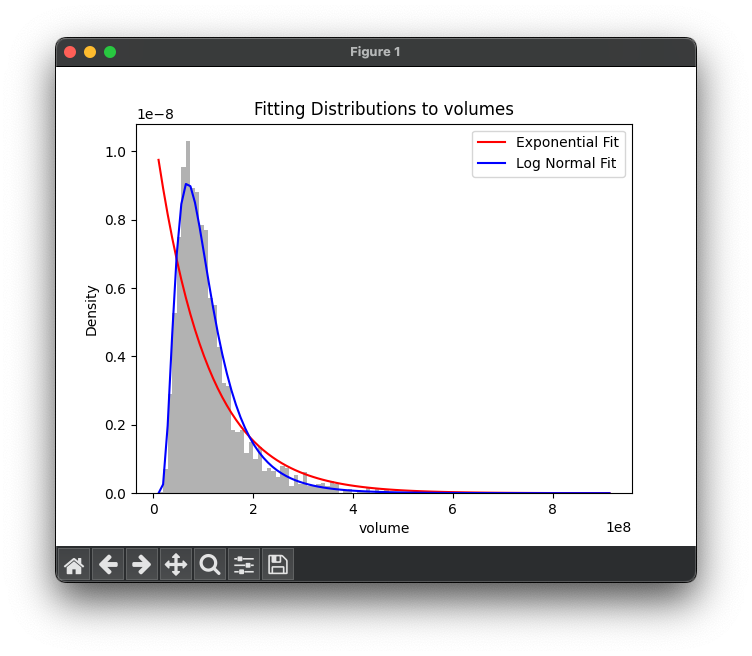
print(f"Chi-square test for Exponential: {chi\_square\_expon}")

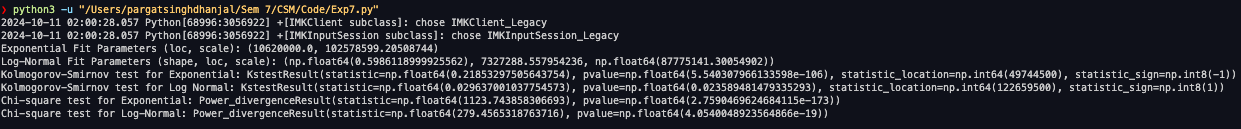
chi\_square\_lognorm = stats.chisquare(*f\_obs*=observed\_freq, *f\_exp*=expected\_freq\_lognorm)

print(f"Chi-square test for Log-Normal: {chi\_square\_lognorm}")

A screenshot of a computer

Description automatically generated





**Conclusion:**

The **log-normal distribution** provides a much better fit for the Tesla stock volumes than the **exponential distribution**, as evidenced by lower Kolmogorov-Smirnov (KS) statistics and chi-square statistics.

However, the **log-normal fit** is a reasonable fit but not perfect. The small p-values from the tests indicate that even the log-normal distribution does not perfectly describe the stock volumes, but it’s closer to the actual data than the exponential distribution.

The large differences found in the **exponential distribution** suggest that it is **not a good fit** for modeling the volume of Tesla stock trades.

**Post lab Questions:**

1. **Explore the concept of multivariate input models. How would you approach input modeling if the input data involved multiple correlated variables?**

Multivariate input modeling involves capturing the relationships between multiple correlated variables. Key steps include exploring correlations (e.g., using a correlation matrix), visualizing relationships (e.g., scatterplot matrix), and selecting appropriate multivariate distributions. Techniques like Principal Component Analysis (PCA) can help reduce dimensionality while preserving the data's structure.