Cory_Golladay_DSC680_Milestone3

July 20, 2025

1 Milestone 3

1.0.1 Create Dataset

[3]:
```
import pandas as pd
import numpy as np

np.random.seed(42)

n = 250
vehicle_types = ['Falcon Heavy', 'Starship', 'SLS']

data = []

for i in range(n):
    payload = np.random.uniform(10, 100)  # in tons
    vehicle = np.random.choice(vehicle_types)
    dev_years = np.random.randint(4, 16)
    launch_year = np.random.randint(2026, 2050)

    # base cost influenced by payload and dev time
    base_cost = payload * 50 + dev_years * 200

    # vehicle multiplier
    if vehicle == 'SLS':
        cost = base_cost * 1.3
    elif vehicle == 'Falcon Heavy':
        cost = base_cost * 1.0
    elif vehicle == 'Starship':
        cost = base_cost * 0.85

    # add noise
    cost += np.random.normal(0, 500)

    data.append([f"M{i+1}", round(payload, 2), vehicle, dev_years, launch_year,␣
↪round(cost, 2)])

df = pd.DataFrame(data, columns=[
```

```
    "Mission_ID", "Payload_Mass_tons", "Vehicle_Type", "Dev_Years",␣
 ↪"Launch_Year", "Est_Cost_Million_USD"
])

# Save as CSV
df.to_csv("mars_missions_dataset.csv", index=False)

df.head()
```

[3]:

```
  Mission_ID  Payload_Mass_tons  Vehicle_Type  Dev_Years  Launch_Year  \
0         M1              43.71  Falcon Heavy         14         2033
1         M2              24.04           SLS         14         2049
2         M3              40.03           SLS          9         2046
3         M4              26.51      Starship          9         2037
4         M5              12.08           SLS         14         2035

   Est_Cost_Million_USD
0               4429.49
1               5362.02
2               4651.76
3               2393.94
4               4299.39
```

[11]:

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

[13]:

```
df = pd.read_csv("mars_missions_dataset.csv")
```

[15]:

```
df_encoded = pd.get_dummies(df, columns=["Vehicle_Type"], drop_first=True)
df_encoded.head()
```

[15]:

```
  Mission_ID  Payload_Mass_tons  Dev_Years  Launch_Year  Est_Cost_Million_USD  \
0         M1              43.71         14         2033               4429.49
1         M2              24.04         14         2049               5362.02
2         M3              40.03          9         2046               4651.76
3         M4              26.51          9         2037               2393.94
4         M5              12.08         14         2035               4299.39

   Vehicle_Type_SLS  Vehicle_Type_Starship
0             False                  False
```

```
1              True                    False
2              True                    False
3             False                     True
4              True                    False
```

[17]:
```
X = df_encoded[["Payload_Mass_tons", "Dev_Years", "Vehicle_Type_Starship",␣
↪"Vehicle_Type_SLS"]]
y = df_encoded["Est_Cost_Million_USD"]
```

[19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[21]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

[21]:
```
LinearRegression()
```

[23]:
```
coef_df = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": model.coef_
})
print("Intercept:", model.intercept_)
coef_df
```

```
Intercept: 75.77247468152473
```

[23]:

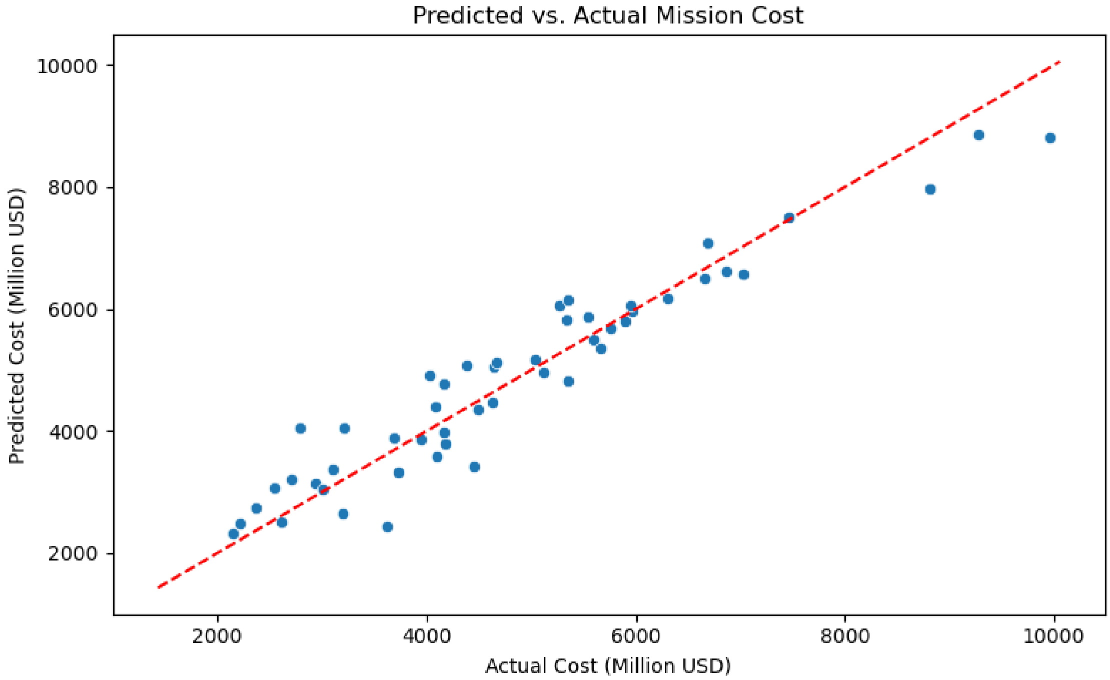
	Feature	Coefficient
0	Payload_Mass_tons	52.899741
1	Dev_Years	183.106571
2	Vehicle_Type_Starship	-786.508994
3	Vehicle_Type_SLS	1344.861377

[25]:
```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"RMSE: ${rmse:,.2f}")
print(f"R² Score: {r2:.4f}")
```

```
RMSE: $519.10
R² Score: 0.9140
```

[29]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel("Actual Cost (Million USD)")
plt.ylabel("Predicted Cost (Million USD)")
```

```
plt.title("Predicted vs. Actual Mission Cost")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.tight_layout()
plt.show()
```

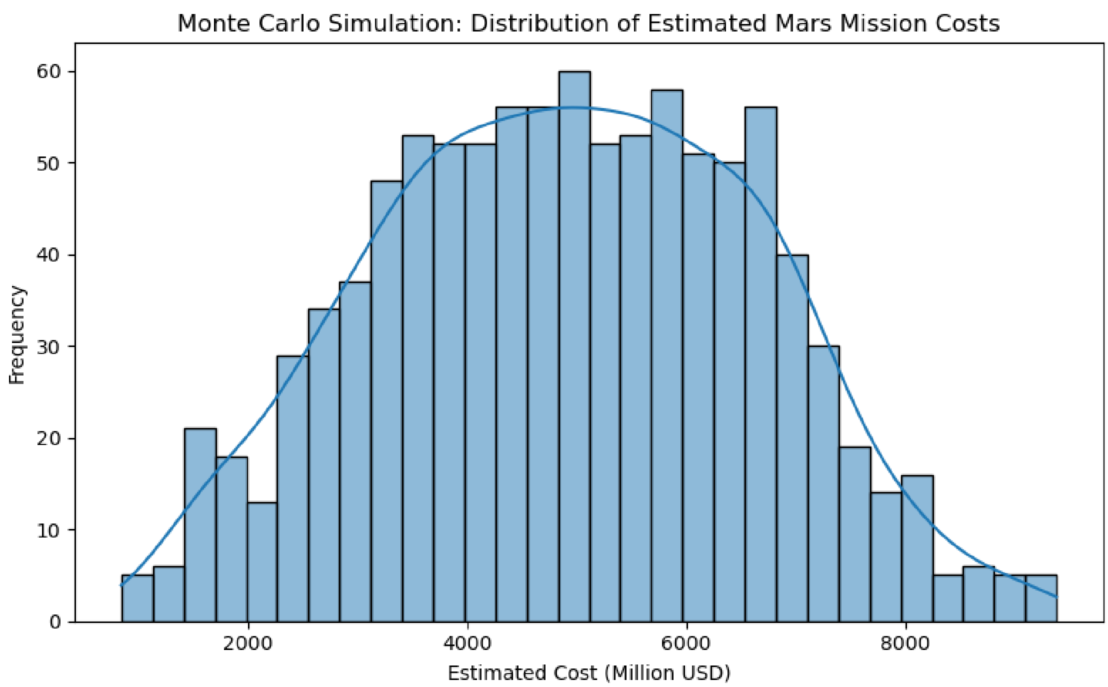


[31]:
```
# Number of simulations
n_simulations = 1000

# Randomly sample features within realistic ranges
sim_payload = np.random.uniform(df["Payload_Mass_tons"].min(),␣
    ↪df["Payload_Mass_tons"].max(), n_simulations)
sim_dev_years = np.random.randint(df["Dev_Years"].min(), df["Dev_Years"].
    ↪max()+1, n_simulations)
sim_vehicle = np.random.choice(["Falcon Heavy", "Starship", "SLS"],␣
    ↪n_simulations)

# Convert vehicle to encoded columns (same as training)
sim_starship = (sim_vehicle == "Starship").astype(int)
sim_sls = (sim_vehicle == "SLS").astype(int)

# Build DataFrame
sim_df = pd.DataFrame({
    "Payload_Mass_tons": sim_payload,
    "Dev_Years": sim_dev_years,
```

```
    "Vehicle_Type_Starship": sim_starship,
    "Vehicle_Type_SLS": sim_sls
})
```

[33]:
```
# Predict cost for each simulated mission
sim_df["Predicted_Cost"] = model.predict(sim_df)
```

[35]:
```
plt.figure(figsize=(8, 5))
sns.histplot(sim_df["Predicted_Cost"], bins=30, kde=True)
plt.title("Monte Carlo Simulation: Distribution of Estimated Mars Mission␣
 ↪Costs")
plt.xlabel("Estimated Cost (Million USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



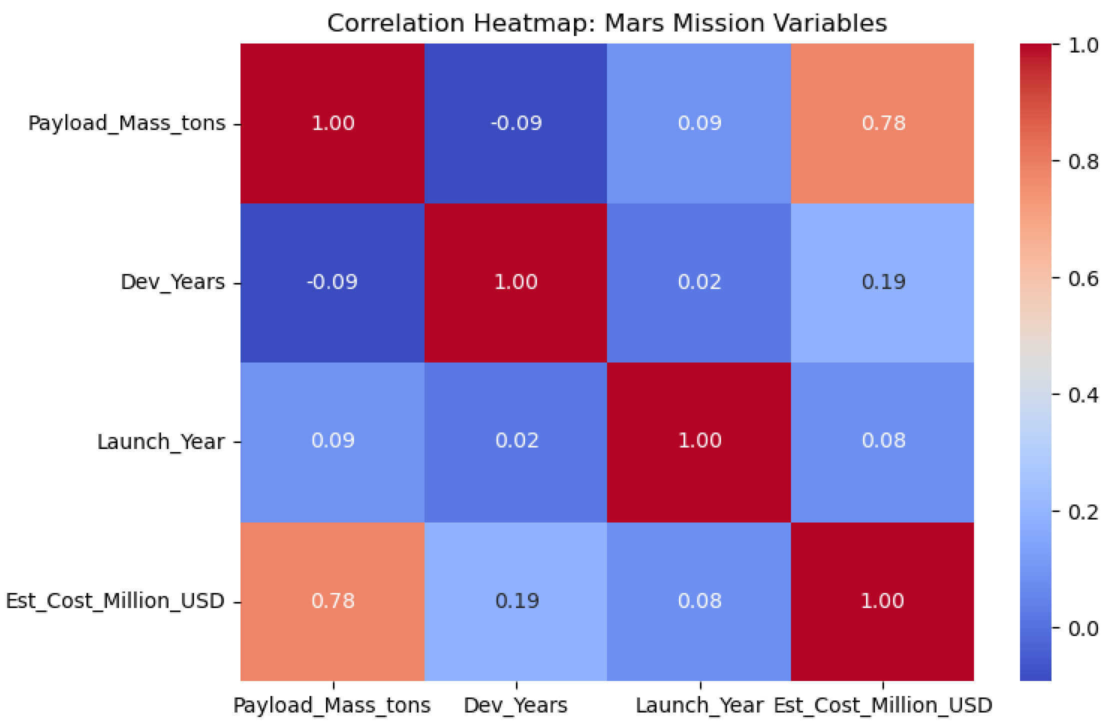
[37]:
```
sim_summary = sim_df["Predicted_Cost"].describe()
sim_summary
```

[37]:
```
count     1000.000000
mean      4914.489184
std       1728.017944
min        848.412481
25%       3641.314041
```

```
50%      4936.838372
75%      6234.331078
max      9382.324253
Name: Predicted_Cost, dtype: float64
```

1.1 Correlation Heatmap

[56]:
```
plt.figure(figsize=(8, 5))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap: Mars Mission Variables")
plt.tight_layout()
plt.savefig("correlation_heatmap.png", dpi=300, bbox_inches="tight")
plt.show()

plt.savefig("CorrelationHeatmap.png", dpi=300, bbox_inches="tight")
```



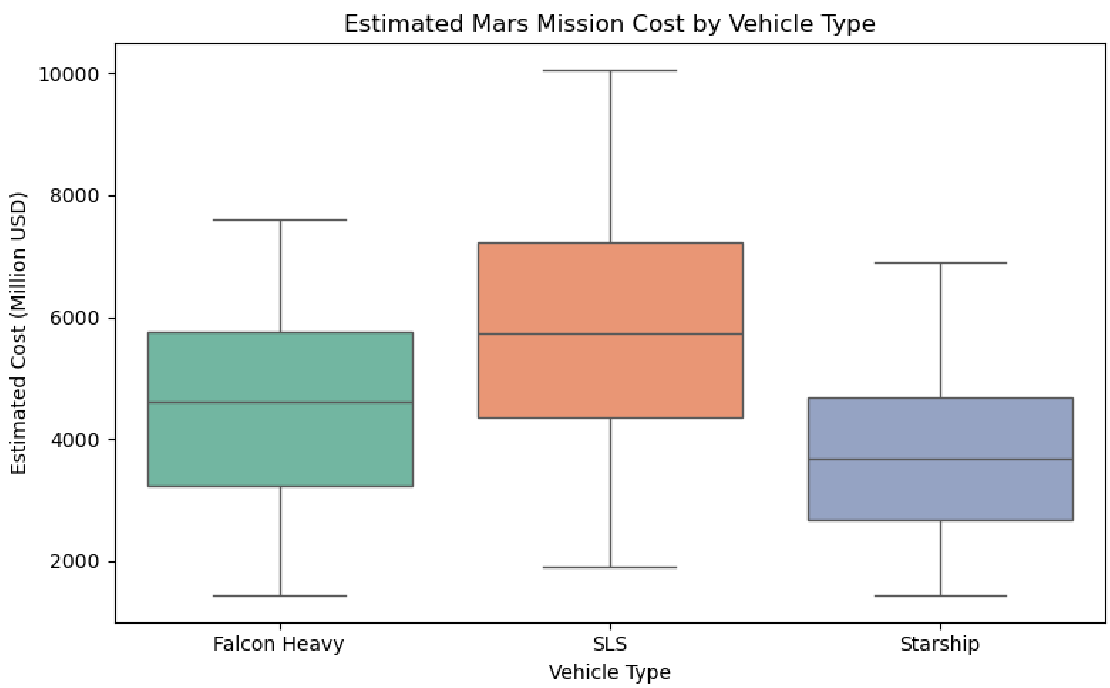

```
<Figure size 640x480 with 0 Axes>
```

1.2 Boxplot: Estimated Cost by Vehicle Type

[58]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x="Vehicle_Type", y="Est_Cost_Million_USD", palette="Set2")
plt.title("Estimated Mars Mission Cost by Vehicle Type")
plt.xlabel("Vehicle Type")
```

```
plt.ylabel("Estimated Cost (Million USD)")
plt.tight_layout()
plt.savefig("cost_by_vehicle_type.png", dpi=300, bbox_inches="tight")
plt.show()

plt.savefig("BoxPlot.png", dpi=300, bbox_inches="tight")
```

```
C:\Users\golla\AppData\Local\Temp\ipykernel_120512\474974977.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=df, x="Vehicle_Type", y="Est_Cost_Million_USD",
palette="Set2")
```

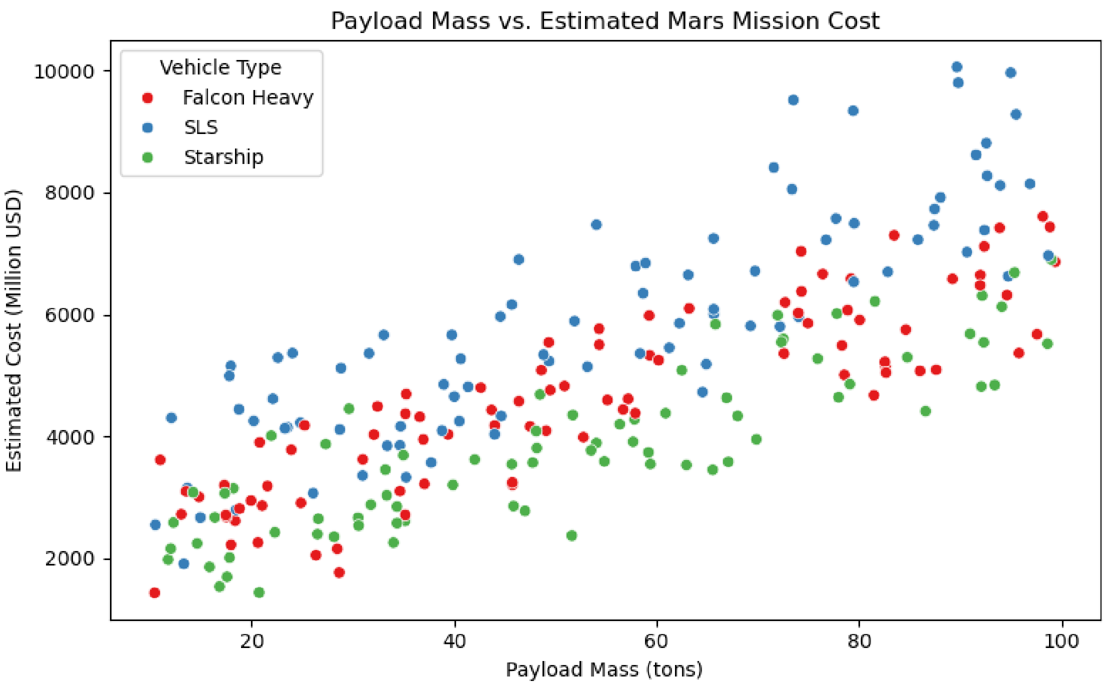



```
<Figure size 640x480 with 0 Axes>
```

1.3 Scatter Plot: Payload Mass vs. Estimated Cost

[60]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x="Payload_Mass_tons", y="Est_Cost_Million_USD",␣
↪hue="Vehicle_Type", palette="Set1")
plt.title("Payload Mass vs. Estimated Mars Mission Cost")
```

```
plt.xlabel("Payload Mass (tons)")
plt.ylabel("Estimated Cost (Million USD)")
plt.legend(title="Vehicle Type")
plt.tight_layout()
plt.savefig("payload_vs_cost.png", dpi=300, bbox_inches="tight")
plt.show()

plt.savefig("ScatterPlot.png", dpi=300, bbox_inches="tight")
```

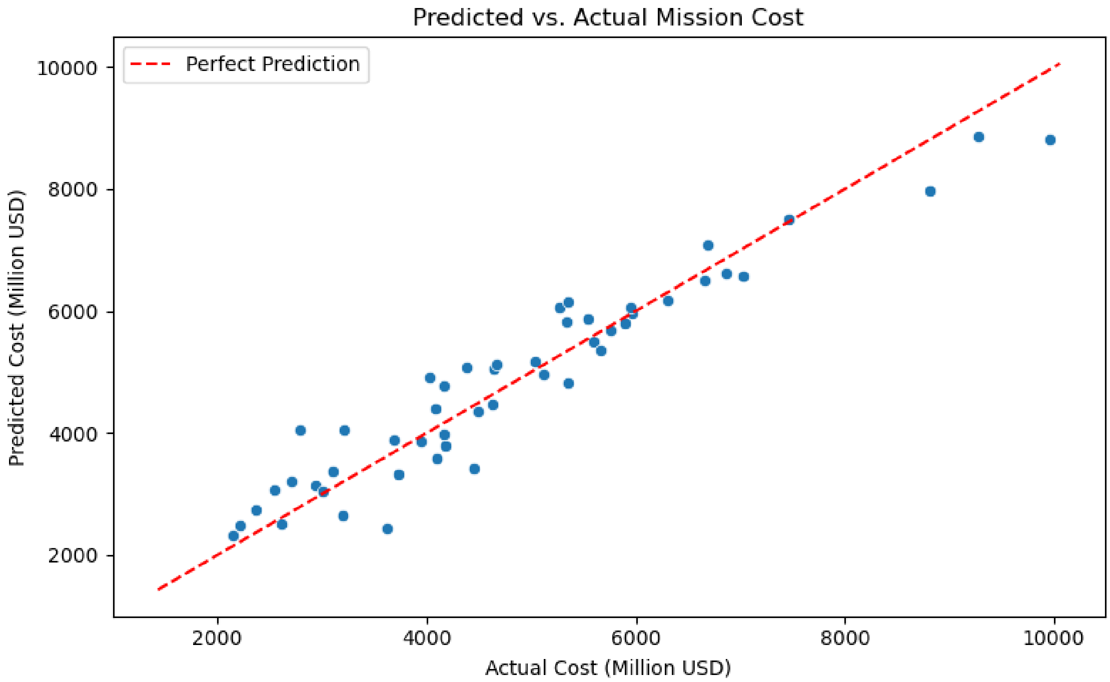



```
<Figure size 640x480 with 0 Axes>
```

1.4 Predicted vs. Actual Mission Cost (Regression Model)

[62]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(x=y_test, y=y_pred)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', label="Perfect␣
↪Prediction")
plt.xlabel("Actual Cost (Million USD)")
plt.ylabel("Predicted Cost (Million USD)")
plt.title("Predicted vs. Actual Mission Cost")
plt.legend()
plt.tight_layout()
plt.savefig("predicted_vs_actual.png", dpi=300, bbox_inches="tight")
plt.show()
```

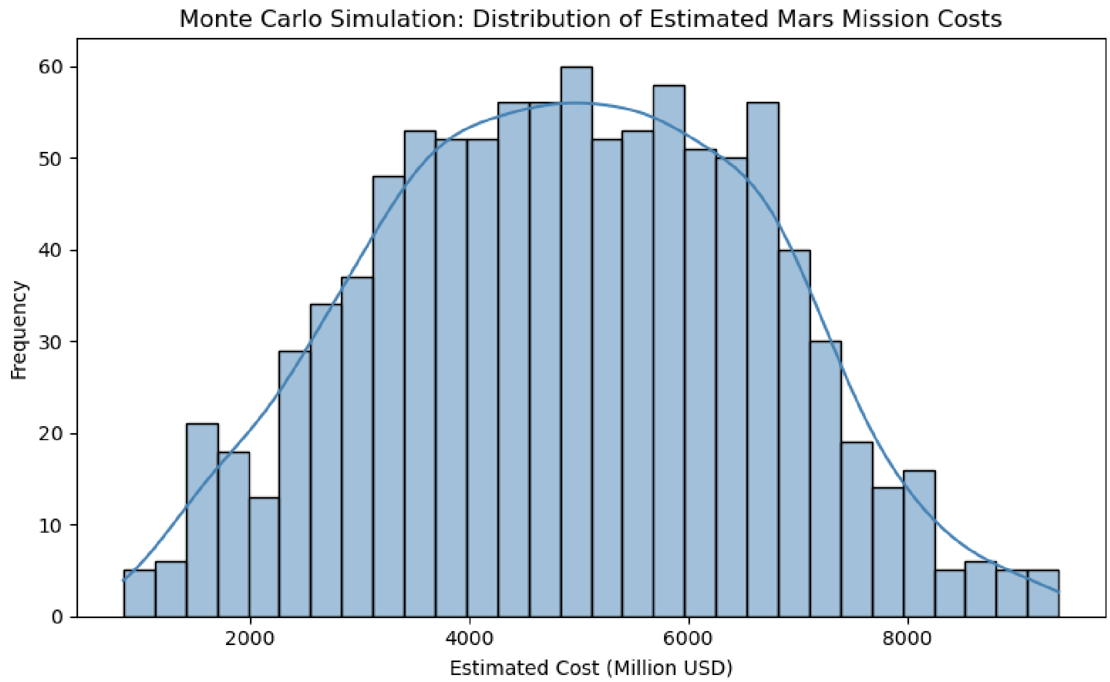
```
plt.savefig("RegressionModel.png", dpi=300, bbox_inches="tight")
```




```
<Figure size 640x480 with 0 Axes>
```

1.5 Monte Carlo Simulation Histogram

[64]:
```
plt.figure(figsize=(8, 5))
sns.histplot(sim_df["Predicted_Cost"], bins=30, kde=True, color="steelblue")
plt.title("Monte Carlo Simulation: Distribution of Estimated Mars Mission␣
↪Costs")
plt.xlabel("Estimated Cost (Million USD)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig("monte_carlo_simulation.png", dpi=300, bbox_inches="tight")
plt.show()

plt.savefig("MonteCarlo.png", dpi=300, bbox_inches="tight")
```



<Figure size 640x480 with 0 Axes>

[ ]: