1. What do you understand about Machine Learning?What are the main stages in the Machine Learning pipeline? Briefly explain the difference between bad data and a bad model.

Machine Learning (ML) is a field of AI where systems learn patterns from data instead of being explicitly programmed. A machine learning model improves its performance automatically as it sees more examples.

Stages in the Machine Learning Pipeline:

1. Data Collection
   Gathering raw data from files, APIs, sensors, etc.

2. Data Cleaning
   Handling missing values, removing duplicates, fixing inconsistencies.

3. Feature Engineering
   Selecting and transforming features so the model can learn better.

4. Model Selection
   Choosing which model to train (regression, decision tree, SVM, etc.).

5. Training the Model
   Feeding data to the model so it learns patterns.

6. Testing & Evaluation
   Checking how well the model performs on unseen data.

7. Deployment
   Sending the model into production (apps, services, systems).

| Bad Data | Bad Model |
|---|---|
| Missing values, noise, wrong labels, outliers | Wrong algorithm choice or poor tuning |
| Garbage-in-garbage-out | Can't learn patterns even with good data |
| Leads to inaccurate learning | Leads to poor predictions even with good data |

2. Why is a regression model used in Machine Learning? Is it a supervised or unsupervised learning method, and why? What are the models for simple and multiple linear regression? Provide the formulas used to calculate the values of betas (no need to derive them) in both cases.

Regression is used in Machine Learning when you want to predict a continuous value. Examples: house price, temperature, salary, medical test value.

Regression is Supervised Learning because the model is trained using labeled data, which is features which are known outputs.

The formulas used to calculate the values of betas in 3 different types of linear regression are given below:

## Types of Linear Regression

### 1. Simple Linear Regression

- One independent variable
- Model:

$$y = \beta_0 + \beta_1 x + \epsilon$$

### Multiple Linear Regression

- Multiple independent variables
- Model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

### Polynomial Regression

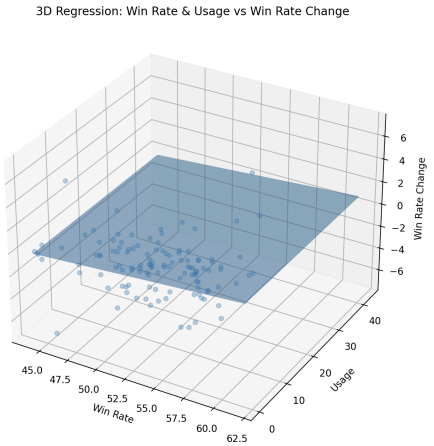
- Extends linear regression by including **non-linear powers of X**
- Equation

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_n X^n$$

3. Download a real-world dataset (not used in class) to train and test a multiple linear regression model. Use it to predict the output. Clearly mention the dataset you used.

a. First, train the model using only two features. Also, create a scatter plot of the dataset using those two features and overlay the regression model. Include screenshots of your code, output, and scatter plot.

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# Load dataset
df = pd.read_csv('/Users/ishan-college/Downloads/clash_royale_cards.csv')

# ---- Select TWO features and ONE target ----
X = df[["Win Rate", "Usage"]]                 # features
y = df["Win Rate Change"]                     # target
# ---------------------------------------------

# Train model
model = LinearRegression()
model.fit(X, y)

print("Intercept (β₀):", model.intercept_)
print("Coefficients (β):", model.coef_)

# 3D Scatter Plot
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X["Win Rate"], X["Usage"], y, alpha=0.3)

# Create prediction plane
x_surf, y_surf = np.meshgrid(
    np.linspace(X["Win Rate"].min(), X["Win Rate"].max(), 50),
    np.linspace(X["Usage"].min(), X["Usage"].max(), 50)
)

z_surf = (
    model.intercept_
    + model.coef_[0] * x_surf
    + model.coef_[1] * y_surf
)

ax.plot_surface(x_surf, y_surf, z_surf, alpha=0.5)

# Labels
ax.set_xlabel("Win Rate")
ax.set_ylabel("Usage")
ax.set_zlabel("Win Rate Change")
ax.set_title("3D Regression: Win Rate & Usage vs Win Rate Change")

plt.show()

# --- User Prediction ---
try:
    win_rate = float(input("Enter Win Rate: "))
    usage = float(input("Enter Usage: "))

    input_df = pd.DataFrame([[win_rate, usage]], columns=["Win Rate", "Usage"])
    prediction = model.predict(input_df)

    print(f"Predicted Win Rate Change: {prediction[0]:.2f}")

except ValueError:
    print("Enter numeric values only.")

```




```
Intercept (β₀): -3.3196465317079062
Coefficients (β): [ 0.07033471 -0.00118427]
Enter Win Rate: 50
Enter Usage: 55
Predicted Win Rate Change: 0.13
```

b. Next, train the model using more than two features of your choice. Show the calculated beta values and use the model to predict the output for some input values.

```python
import pandas as pd
from sklearn.linear_model import LinearRegression

# Load dataset
df = pd.read_csv('/Users/ishan-college/Downloads/clash_royale_cards.csv')

# ---- Select MORE THAN 2 features ----
X = df[["Win Rate", "Usage", "Usage Change"]]

# Select target variable
y = df["Win Rate Change"]
# -------------------------------------

# Train model
model = LinearRegression()
model.fit(X, y)

# Print intercept and coefficients
print("Intercept (β₀):", model.intercept_)
print("\nCoefficients (β values):")
for feature, coef in zip(X.columns, model.coef_):
    print(f"{feature}: {coef}")

# ---- Make Predictions ----
print("\n--- Prediction Example ---")

try:
    win_rate = float(input("Enter Win Rate: "))
    usage = float(input("Enter Usage: "))
    usage_change = float(input("Enter Usage Change: "))

    input_df = pd.DataFrame([[win_rate, usage, usage_change]],
                            columns=["Win Rate", "Usage", "Usage Change"])

    prediction = model.predict(input_df)

    print(f"\nPredicted Win Rate Change: {prediction[0]:.4f}")

except ValueError:
    print("Invalid input. Please enter numeric values.")
```

```
Intercept (β₀): -2.3304773484490116

Coefficients (β values):
Win Rate: 0.05133393960423274
Usage: 0.0055068847975453476
Usage Change: 0.5892472048789782

--- Prediction Example ---
Enter Win Rate: 5
Enter Usage: 7
Enter Usage Change: 1

Predicted Win Rate Change: -1.4460
```

For this dataset, I used a real-world Clash Royale balance dataset named: clash_royale_cards.csv from [Clash Royale Cards Stats - Kaggle]