1. How can we categorize a search algorithm in AI as informed or uninformed? Why is the Uniform Cost Search algorithm considered uninformed (blind)? Do you think the solution provided by Greedy Best-First Search is always optimal? Why or why not?

Informed Search (Heuristic Search): Uses heuristics, or shortcuts that help it understand the situation, to figure out which branch of the tree to look at next. This help usually makes looking for information or files a lot faster and easier.

Examples: A*, Greedy Best-First Search

Uninformed Search (Blind Search): Has no extra information about what's supposed to happen at the end other than what was given in the problem's description. It goes through all the possible solutions without really knowing which ones are actually good or not.

Examples: Breadth-First Search first looks at the nodes closest to the start node, then keeps moving outward. Depth-First Search looks at the nodes deepest away from the starting node and works its way up toward the start. Uniform Cost Search uses an extended version of Depth-First Search and is mainly used for search problems where the cost of moving between nodes is not the same.

Ans:-

- In UCS, the only way to measure the path is by looking at its cost (g(n)), and no estimate of the goal is taken into account.

- It doesn't take into account how far away the goal is, so it acts like an uninformed or blind strategy.

Ans:-

- In some cases, it does not lead to the best outcome.

- Whenever selecting the path, it only considers the heuristic cost (h(n)) and ignores the path cost (g(n)).

- Using an overestimating heuristic may result in solutions that are not the best possible.

2. What do you understand by Machine Learning? How is it different from traditional programming, and why is it important?

Ans:-
Machine Learning (ML) is focused on getting computers to learn without being given set instructions.

ML algorithms analyze data to spot patterns and use them to predict or make decisions.

Traditional Programming	Machine Learning
Programmer writes explicit rules	Algorithm learns rules from data
Input + Rules → Output	Input + Output → Rules (Model)
Static and deterministic	Adaptive and data-driven

Importance of ML:

Manages difficult issues such as speech recognition and fraud detection

Scales well with data

It becomes better at its job as more data is added (learning from previous experience)

3. What are the main stages in the Machine Learning pipeline?

- Collecting the necessary information.
- Data cleaning, imputation, standardization, scaling, etc.
- Feature Selection and Engineering – Selecting or building features that better represent the data.
- Selecting the Most Suitable Algorithm for the Task.
- The algorithm is fed data in order to learn patterns from it.
- Evaluating Models – Analyzing using metrics like accuracy, precision, etc.
- Finding the optimal values for hyperparameters to improve model performance.

4. What are the main challenges in Machine Learning? Briefly explain the difference between bad data and a bad model.

- Overfitting and Underfitting
- Bias-Variance Tradeoff
- Data Quality and Quantity
- Computational Resources
- Interpretability of Models
- Ethical Concerns (e.g., bias, privacy)

Bad Data vs. Bad Model:

- Bad Data: Noisy, incomplete, inconsistent, irrelevant data that misleads the model.
    - Example: Mislabeled training data
- Bad Model: Incorrect or inappropriate algorithm, poor tuning, or insufficient learning.
    - Example: Using linear regression for non-linear data

5. 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.

- Purpose: Regression models predict a continuous output variable based on one or more input features.
- Type: Supervised Learning, because it learns from labeled data (input-output pairs).

Simple Linear Regression:

$$y=\beta 0+\beta 1x+\epsilon$$

Multiple Linear Regression Model:

$$y=\beta 0+\beta 1x1+\beta 2x2+\cdots+\beta nxn+\epsilon$$

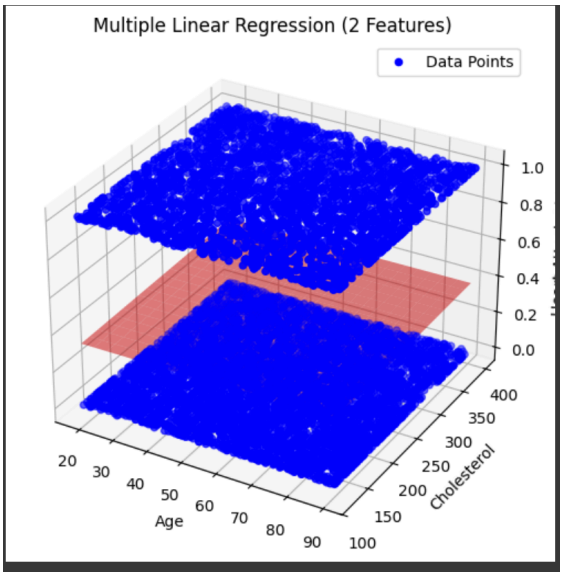
Multiple Linear Regression

$$\beta=(XTX)-1XTy$$

6. 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
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# Load dataset
df = pd.read_csv("/content/heart_attack_prediction_dataset.csv")

# Select and clean data
df = df[['Age', 'Cholesterol', 'Heart Attack Risk']].dropna()

# Prepare X and y
X = df[['Age', 'Cholesterol']]
y = df['Heart Attack Risk']

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

# Predict plane for plotting
age_vals = np.linspace(X['Age'].min(), X['Age'].max(), 20)
chol_vals = np.linspace(X['Cholesterol'].min(), X['Cholesterol'].max(), 20)
age_grid, chol_grid = np.meshgrid(age_vals, chol_vals)
Z = model.predict(np.c_[age_grid.ravel(), chol_grid.ravel()]).reshape(age_grid.shape)

# 3D Plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X['Age'], X['Cholesterol'], y, c='blue', label='Data Points')
ax.plot_surface(age_grid, chol_grid, Z, alpha=0.5, color='red')
ax.set_xlabel("Age")
ax.set_ylabel("Cholesterol")
ax.set_zlabel("Heart Attack Risk")
ax.set_title("Multiple Linear Regression (2 Features)")
plt.legend()
plt.show()
```



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
# Select features
features = ['Age', 'Cholesterol', 'Heart Rate', 'BMI', 'Triglycerides']

# Reload the dataset to ensure all columns are available for the second model
df = pd.read_csv("/content/heart_attack_prediction_dataset.csv")

# Select the required columns and drop rows with missing values
df = df[features + ['Heart Attack Risk']].dropna()

# Train model
X = df[features]
y = df['Heart Attack Risk']
model = LinearRegression()
model.fit(X, y)

# Output beta values
print("Intercept:", model.intercept_)
for feature, coef in zip(features, model.coef_):
    print(f"{feature}: {coef}")

# Predict sample values
sample = pd.DataFrame({
    'Age': [45, 60],
    'Cholesterol': [210, 330],
    'Heart Rate': [80, 95],
    'BMI': [28, 35],
    'Triglycerides': [250, 400]
})
predictions = model.predict(sample)
print("Predictions:", predictions)
```

```
Intercept: 0.3189147118983765
Age: 0.0001472811992937321 5
Cholesterol: 0.00011540560740274352
Heart Rate: -0.00010175206825152757
BMI: -1.6150607464076033e-05
Triglycerides: 2.2732179398171975e-05
Predictions: [0.34686821 0.36469659]
```