4.2. Experiment No. 1

Aim: Feature Transformation

A) To use PCA Algorithm for dimensionality reduction. You have a dataset that includes measurements for different variables on wine (alcohol, ash, magnesium, and so on). Apply PCA algorithm & transform this data so that most variations in the measurements of the variables are captured by a small number of principal components so that it is easier to distinguish between red and white wine by inspecting these principal components. Dataset Link: https://media.geeksforgeeks.org/wp-content/uploads/Wine.csv

Objective:

The main objectives of this experiment are to:

1. Understand and implement the PCA algorithm for dimensionality reduction.
2. Transform the wine dataset to a lower-dimensional space using PCA.
3. Visualize the transformed data to observe patterns and differences between red and white wines.

Theory:

Principal Component Analysis (PCA): Principal Component Analysis (PCA) is a dimensionality reduction technique used to reduce a large set of variables to a smaller one that still contains most of the important information (variance) in the data.

Goals of PCA:

- Reduce dimensionality.
- Retain as much variance (information) as possible.
- Find new axes (components) that are linear combinations of original features.
- Remove correlation (new components are orthogonal).

When to Use PCA:

- You have high-dimensional data.
- Many features are correlated.
- You want to visualize data in 2D/3D.
- To improve training speed and accuracy by eliminating noise.

Example Dataset (2D):

Let's say we have the following 4 data points with two features:

X (Height in cm) Y (Weight in kg)
160 55
165 65
170 60
175 70

We want to reduce this to 1D using PCA.

Step 1: Standardize the data

PCA is sensitive to scale, so we standardize (mean=0, std=1) each feature.
Let's compute the mean and standard deviation for each column:
Mean height = (160 + 165 + 170 + 175) / 4 = 167.5
Mean weight = (55 + 65 + 60 + 70) / 4 = 62.5
Now subtract mean and divide by std:

Standardized X (Height)	**Standardized Y (Weight)**
(160 - 167.5) = -7.5	(55 - 62.5) = -7.5
-2.5	2.5
2.5	-2.5
7.5	7.5

Then divide by standard deviation (assume std = 6.45 for both):

Zx	**Zy**
-1.16	-1.16
-0.39	0.39
0.39	-0.39
1.16	1.16

Step 2: Compute the covariance matrix

The covariance matrix tells us how the features vary with each other:

$$\text{Cov}(X,Y) - \frac{1}{n-1}\sum_{i-1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

For standardized data:

$$\text{Cov matrix} - \begin{bmatrix} 1 & 0.93 \\ 0.93 & 1 \end{bmatrix}$$

(Here, correlation is high — this means the features are not independent.)

Step 3: Compute eigenvectors and eigenvalues

These tell us:

- **Eigenvectors**: Directions of new axes (principal components).
- **Eigenvalues**: How much variance each axis captures.

For this matrix, suppose we get:

- **PC1 (first eigenvector)** = [0.71, 0.71], eigenvalue = 1.93
- **PC2 (second eigenvector)** = [-0.71, 0.71], eigenvalue = 0.07

Interpretation:

- PC1 captures **most variance** (~96%)
- PC2 captures **little variance** (~4%)

Step 4: Project data onto principal components
Now we project original standardized data onto the **PC1 axis**:

Use dot product:
For point [-1.16, -1.16] and PC1 = [0.71, 0.71]

Z=[−1.16,−1.16]·[0.71,0.71]=−1.64Z = [-1.16, -1.16] \cdot [0.71, 0.71] = -1.64Z=[−1.16,−1.16]·[0.71,0.71]=−1.64

Do this for all points, and now you have a **1D representation** that captures the most important information in your data.

Applications:

PCA has applications in various domains:

- Face recognition (Eigenfaces)
- Data compression (e.g., image storage)

- Noise reduction (e.g., EEG data)
- Exploratory data analysis (visualization in 2D/3D)
- Preprocessing before training machine learning models

Input:

The input for this experiment is the wine dataset, which includes measurements for different variables related to wine characteristics.

Output:

The output of this experiment includes:

1. Transformed dataset with reduced dimensions using PCA.
2. Visualization of the transformed data to observe the distinction between red and white wines.

Conclusion:

Through the application of PCA, we successfully demonstrated how to transform high-dimensional wine data into a lower-dimensional space while retaining important variations. This experiment aids in understanding how PCA can assist in distinguishing between different types of wines based on principal components.

Outcome:

The outcome of this experiment is a transformed dataset with reduced dimensions using PCA, as well as visualizations that help observe the separation between red and white wines in the transformed space. This showcases the effectiveness of PCA in simplifying data representation while preserving relevant information.

Questions:

1. What is the primary objective of applying the PCA algorithm in this experiment on the wine dataset? How does PCA achieve dimensionality reduction while retaining important variations in the data?
2. Why is it important to distinguish between red and white wines based on principal components? How can PCA help in revealing patterns that differentiate these wine types?
3. Describe the process of transforming the wine dataset using PCA. What are the steps involved in calculating the principal components and projecting the data onto them?
4. After applying PCA and obtaining the transformed data, how can you visualize the separation between red and white wines? What kind of plot or visualization technique might be used to achieve this?
5. What are some potential advantages and limitations of using PCA for feature transformation? How might the choice of the number of principal components impact the interpretability and performance of the transformed data in classification tasks?

4.3. Experiment No. 2

Aim: Regression Analysis

A) Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks:

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression and ridge, Lasso regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset

Objective:

The main objectives of this experiment are to:

1. Preprocess the dataset to make it suitable for regression analysis.
2. Detect and handle outliers that might affect the predictive models.
3. Assess the correlation between features to understand relationships.
4. Apply linear regression, ridge regression, and Lasso regression models for price prediction.
5. Evaluate and compare the performance of the models using metrics such as R2 and RMSE.

Theory:

Regression Analysis is a statistical method used to model the **relationship between a dependent variable (target)** and one or more **independent variables (features)**.

It allows us to:

- **Predict** values (e.g., price, temperature, sales)
- **Understand** the strength and nature of relationships
- **Quantify** the impact of each feature on the output

Type	When to Use
Linear Regression	Linear relationship between variables
Multiple Linear Regression	More than one independent variable
Polynomial Regression	Relationship is nonlinear but can be modeled by powers
Ridge / Lasso Regression	To prevent overfitting in complex models
Logistic Regression	Binary classification (Yes/No, 0/1) — despite the name, it's not used for continuous prediction

Linear Regression: The Basics

Equation of a Line:

$$y = mx + b$$

Where:

- y = predicted value (dependent variable)
- x = input feature (independent variable)
- m = slope (how much y changes when x changes)
- b = intercept (value of y when x = 0)

General Linear Regression Formula:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

- β_0 is the intercept
- β_n are the coefficients
- ε is the error term (residual)

Predict the weight of a person based on their height using linear regression.

Dataset:

Height (cm)	Weight (kg)
150	52
160	58
170	65
180	70
190	78

✅ **Step 1: Plot the Data**

Before anything, visualize it.

You'd likely see a trend: as height increases, weight also increases — this suggests a positive linear relationship.

✅ **Step 2: Fit a Linear Regression Model**

We want to fit a line of best fit:

$$\text{Weight} = m \times \text{Height} + b$$

Let's say after calculating, we find:

$$\text{Weight} = 0.6 \times \text{Height} - 40$$

✅ **Step 3: Use the Model to Predict**

If someone is 175 cm tall:

$$\text{Weight} = 0.6 \times 175 - 40 = 105 - 40 = 65 \text{ kg}$$

So the predicted weight is **65 kg**.

Common evaluation metrics:

Metric	Meaning
R^2 Score (Coefficient of Determination)	% of variance explained by model
MAE (Mean Absolute Error)	Avg absolute error
MSE (Mean Squared Error)	Avg squared error (punishes large errors)
RMSE	Square root of MSE

Suppose:

- $R^2 = 0.97 \rightarrow$ Model explains 97% of the variation.
- RMSE = 2.5 kg → Predictions deviate on average by 2.5 kg.

That's a good model.

Applications:

Use Case	Independent Variables (Features)	Dependent Variable
House price prediction	Area, bedrooms, location	Price
Weather forecast	Humidity, wind speed, season	Temperature
Uber fare estimate	Distance, duration, time of day	Fare

Input:
The input for this experiment is the Uber ride price dataset, which contains features related to pickup and drop-off locations, time, and ride attributes.

Output:

The output of this experiment includes:

1. Trained regression models (linear, ridge, and Lasso) for predicting Uber ride prices.
2. Evaluation metrics (R2, RMSE) for each model, facilitating comparison.

Conclusion:

Through the application of regression analysis techniques, we successfully demonstrated the prediction of Uber ride prices based on provided features. The experiment highlighted the effectiveness of different regression models and their performance evaluation.

Outcome:

The outcome of this experiment is a set of trained regression models (linear, ridge, and Lasso) capable of predicting Uber ride prices. Additionally, we obtain insights into the relative performance of these models using evaluation metrics.

Questions:

1. What is the primary goal of regression analysis in the context of predicting Uber ride prices, and how does it differ from classification analysis?
2. How can outliers in the dataset potentially impact the accuracy and reliability of regression models? What techniques can be used to identify and handle outliers?
3. Explain the concept of correlation in the context of feature analysis for regression. How does understanding correlation help in feature selection and model building?
4. Describe the key differences between linear regression, ridge regression, and Lasso regression. How does each technique address the issue of overfitting in regression models?
5. In the context of evaluating regression models, what do R2 (coefficient of determination) and RMSE (root mean squared error) represent? How can these metrics be used to compare and select the best-performing model?

4.4. Experiment No. 3

Aim: Classification Analysis

A) Implementation of Support Vector Machines (SVM) for classifying images of hand-written digits into their respective numerical classes (0 to 9).

Objective:

1. Implement a Support Vector Machine classifier for recognizing handwritten digits.
2. Train the SVM model on a dataset of labeled images to learn patterns and characteristics of different digits.
3. Evaluate the performance of the trained SVM model in terms of classification accuracy.

Theory:

Support Vector Machine (SVM) is a supervised learning algorithm used for **classification and regression**.

Goal of SVM:

Find a **hyperplane** that best separates data into classes with the **maximum margin** between the closest points (support vectors).

For non-linearly separable data, SVM uses **kernels** to project data into higher dimensions.

Key Concepts Behind SVM:

a) Hyperplane

A hyperplane is a **line (in 2D)**, **plane (in 3D)**, or **flat decision boundary** that separates classes.

In 2D, it's:

$$w_1 x_1 + w_2 x_2 + b = 0$$

- One side of the hyperplane: class 1
- Other side: class 2

---

b) Margin

The **margin** is the distance between the **hyperplane** and the **closest data points** from each class.

- **Goal of SVM:** Maximize this margin.
- Why? Larger margins lead to **better generalization** (less overfitting).

c) Support Vectors

The **data points that lie closest** to the decision boundary (margin edges).
They are critical — moving them would **change the position of the hyperplane**.

How SVM Works: Step-by-Step

Step 1: Input data

You have labeled data: each sample belongs to class A or B.

Step 2: Find a hyperplane

SVM tries to find the **hyperplane** that:

- Clearly separates the classes
- Maximizes the **margin** between them

Step 3: Optimize

SVM solves an optimization problem to **maximize margin**, ensuring the best separator.

What if Data is Not Linearly Separable?

- Real-world data often looks like this:
- SVM handles this using kernels.

What Are Kernels in SVM?

Kernel trick allows SVM to operate in higher-dimensional space without computing the transformation explicitly.

It helps SVM classify data that's not linearly separable by mapping it to a space where it is.

Common Kernel Types:

Kernel	Use Case	Equation
Linear	Data is linearly separable	$K(x, x') = x \cdot x'$
Polynomial	Non-linear with polynomial features	$(x \cdot x' + c)^d$
RBF (Gaussian)	Default for most cases	$\exp(-\gamma\|x - x'\|^2)$
Sigmoid	Similar to neural nets	$\tanh(\alpha x \cdot x' + c)$

Why to Use SVM for Handwritten Digit Classification?

- SVM is powerful with **high-dimensional data** (e.g., images → pixels)
- Handles **non-linear boundaries** using kernels
- Works well on **small to medium-sized** datasets (like MNIST or digits)

Applications:

1. Handwriting Recognition: As in this experiment, SVMs can be used to recognize and classify handwritten characters or digits in various applications.
2. Image Classification: SVMs have been used for classifying objects, scenes, or patterns in images.
3. Medical Diagnosis: SVMs can assist in diagnosing diseases based on medical image analysis.
4. Text and Document Classification: They are used in sentiment analysis, spam filtering, and categorizing text documents.

Input:

The input for this experiment is a dataset of images containing handwritten digits (0 to 9). Each image is labeled with the corresponding digit it represents.

Output:

The output of this experiment includes:

1. A trained SVM model capable of classifying new handwritten digit images.
2. Classification performance metrics, such as accuracy, precision, recall, and F1-score.

Conclusion:

Through the implementation of Support Vector Machines, we successfully demonstrated the ability to classify handwritten digits into their respective numerical classes. This experiment showcases the versatility of SVMs in image classification tasks.

Outcome:

The outcome of this experiment is a trained SVM model that can accurately classify handwritten digits. This model can be used to automatically recognize and classify handwritten characters in various applications.

Questions:

1. How does a Support Vector Machine (SVM) classify handwritten digits into numerical classes, and what is the key principle behind its classification approach?
2. What is the purpose of the kernel trick in SVMs, especially in the context of classifying handwritten digits? How does it enable SVMs to handle nonlinearly separable data?
3. In the context of this experiment, what are some metrics commonly used to evaluate the performance of the SVM model in classifying handwritten digits? Explain each metric briefly.
4. How can SVMs handle multi-class classification, considering that handwritten digits can belong to one of ten classes (0 to 9)? What techniques are typically used to extend SVMs to multi-class scenarios?
5. What are some potential challenges or limitations that might be encountered when using SVMs for handwritten digit classification? How might these challenges be addressed or mitigated?

4.5. Experiment No. 4

Aim: Clustering Analysis

A) Implement K-Means clustering on Iris.csv dataset. Determine the number of clusters using the elbow method.

Objective:

The main objective of this experiment is to understand and implement K-Means clustering and the elbow method for selecting the appropriate number of clusters in the given Iris dataset.

Theory:

K-Means Clustering is an **unsupervised learning algorithm** used to **group similar data points** into **K distinct, non-overlapping clusters**, based on **feature similarity**.

It tries to **minimize the distance** between points within a cluster and **maximize the distance** between different clusters.

How K-Means Works — Step-by-Step

1. **Choose K** (the number of clusters).
2. **Initialize K centroids** randomly.
3. **Assign** each point to the nearest centroid (based on Euclidean distance).
4. **Recalculate** centroids as the average of points in each cluster.
5. **Repeat** steps 3–4 until:
    - Centroids stop moving (convergence), or
    - A max number of iterations is reached.

Example: Grouping 6 Points into 2 Clusters

Sample Data (x, y coordinates):

Point	x	y
A	1	2
B	1	4
C	1	0
D	10	2
E	10	4
F	10	0

We can clearly see:

- Points A, B, C are near (1, *)
- Points D, E, F are near (10, *)

We expect two clusters:

- Cluster 1: A, B, C
- Cluster 2: D, E, F

Point	Coordinates (x, y)
A	(1, 2)
B	(1, 4)
C	(1, 0)
D	(10, 2)
E	(10, 4)
F	(10, 0)

Step 0: Choose K = 2 clusters and initialize centroids
We randomly pick two initial centroids. Let's choose:
Centroid 1 (C1): Point A → (1, 2)
Centroid 2 (C2): Point D → (10, 2)

Step 1: Assign each point to nearest centroid by Euclidean distance

Euclidean distance between two points (x_1, y_1) and (x_2, y_2):

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Point	Dist to C1=(1,2)	Dist to C2=(10,2)	Assigned Cluster
A (1,2)	0	$\sqrt{(10-1)^2 + (2-2)^2} = 9$	C1 (cluster 1)
B (1,4)	$\sqrt{(1-1)^2 + (4-2)^2} = 2$	$\sqrt{(10-1)^2 + (4-2)^2} = \sqrt{81+4} = 9.22$	C1
C (1,0)	$\sqrt{(1-1)^2 + (0-2)^2} = 2$	$\sqrt{(10-1)^2 + (0-2)^2} = \sqrt{81+4} = 9.22$	C1
D (10,2)	9	0	C2 (cluster 2)
E (10,4)	$\sqrt{(10-1)^2 + (4-2)^2} = 9.22$	$\sqrt{(10-10)^2 + (4-2)^2} = 2$	C2
F (10,0)	$\sqrt{(10-1)^2 + (0-2)^2} = 9.22$	$\sqrt{(10-10)^2 + (0-2)^2} = 2$	C2

Step 2: Calculate new centroids

Calculate centroid coordinates by averaging the points in each cluster.

- Cluster 1 points: A(1,2), B(1,4), C(1,0)

$$C1_{new} = \left(\frac{1+1+1}{3}, \frac{2+4+0}{3}\right) = (1, 2)$$

- Cluster 2 points: D(10,2), E(10,4), F(10,0)

$$C2_{new} = \left(\frac{10+10+10}{3}, \frac{2+4+0}{3}\right) = (10, 2)$$

Notice: Centroids did not move in this iteration!

Step 3: Check for convergence

Since centroids didn't change, the algorithm converges.

Iteration	Centroid 1	Centroid 2	Cluster 1 Points	Cluster 2 Points
Init	(1, 2)	(10, 2)	—	—
1	(1, 2)	(10, 2)	A(1,2), B(1,4), C(1,0)	D(10,2), E(10,4), F(10,0)

Applications:

Clustering has applications in various domains:

1. **Customer Segmentation:** Identifying distinct customer groups based on purchasing behaviour.
2. **Image Segmentation:** Grouping similar regions in images for analysis or compression.
3. **Anomaly Detection:** Identifying unusual patterns in data by identifying clusters with fewer points.
4. **Document Clustering:** Grouping similar documents for information retrieval.
5. **Genomic Clustering:** Identifying patterns in DNA sequences for genetic research.

Input:

The input for this experiment is the Iris.csv dataset, which contains measurements of iris flowers' features (sepal length, sepal width, petal length, and petal width).

Output:

The output of this experiment includes:

1. Clusters: The identified clusters of data points after applying the K-Means algorithm.
2. Optimal Number of Clusters: The number of clusters determined using the elbow method.

Conclusion:

Through the implementation of K-Means clustering on the Iris dataset, we successfully demonstrated how this technique can group similar iris flowers based on their measurements. The elbow method allowed us to determine the optimal number of clusters for the given data.

Outcome:

The outcome of this experiment is a set of clusters formed through K-Means clustering, along with the optimal number of clusters identified using the elbow method. This showcases the ability to organize and analyze data without any labeled information.

Questions:

1. What is the primary goal of K-Means clustering, and how does it partition a dataset into clusters? Briefly explain the iterative process of K-Means.
2. Describe the elbow method for determining the optimal number of clusters in K-Means clustering. How does the sum of squared distances play a role in this technique?
3. In the context of the Iris dataset, what are the features being considered for clustering? How might the choice of features affect the clustering results?
4. How does the concept of within-cluster sum of squares (WCSS) relate to the elbow method? How does the plot of WCSS values help in identifying the optimal number of clusters?
5. What are some potential challenges or limitations when using the elbow method to determine the number of clusters? Are there scenarios where the elbow method might not provide a clear-cut solution?

4.6.Experiment No. 5

Aim: Ensemble Learning

A) Implement Random Forest Classifier model to predict the safety of the car. Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set

Objective:

The main objective of this experiment is to utilize ensemble learning techniques, specifically the Random Forest Classifier, to create a predictive model that can assess the safety of cars based on relevant features.

Theory:

Random Forest Classifier:

Random Forest is an **ensemble learning** method used for classification (and regression). It builds **multiple decision trees** during training and outputs the **mode (majority vote)** of the classes predicted by individual trees.

Why Random Forest?

- Helps reduce **overfitting** compared to a single decision tree.
- Handles **high dimensional data** well.
- Robust to **noise** and **outliers**.
- Works well with **large datasets**.

How it works?

1. **Bootstrap Sampling**: Randomly sample data points with replacement to create different subsets.
2. **Decision Trees**: Train a decision tree on each subset.
3. **Random Feature Selection**: At each split in a tree, only a random subset of features is considered.
4. **Aggregation**: For classification, predictions from all trees are combined by majority voting.

Applications:

Ensemble Learning has applications in various domains:

1. Finance: Credit risk assessment and fraud detection.
2. Medicine: Disease prediction and patient diagnosis.
3. Natural Language Processing: Sentiment analysis and text classification.
4. Image Processing: Object detection and recognition.
5. Autonomous Vehicles: Scene interpretation and obstacle avoidance.

Input:

The input for this experiment includes the dataset provided in the link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set. The dataset contains features related to car specifications and attributes.

Output:

The output of this experiment is a predictive model built using the Random Forest Classifier. For each car in the dataset, the model will provide a prediction of its safety level, which could be a categorical label indicating low, moderate, or high safety.

Conclusion:

Through the implementation of the Random Forest Classifier on the car safety dataset, we successfully demonstrated how ensemble learning can be used to predict the safety levels of cars. The experiment showcases the power of combining multiple decision trees to create a robust and accurate predictive model.

Outcome:

The outcome of this experiment is a trained Random Forest Classifier model that can predict the safety level of cars with a certain degree of accuracy. This model can be used to evaluate the safety of new cars and aid decision-making in the automotive industry.

Questions:

1. How does ensemble learning, specifically the Random Forest Classifier, contribute to improving the accuracy and robustness of predictive models compared to individual decision trees?
2. What are some key advantages of using the Random Forest Classifier for predicting car safety based on the provided dataset?
3. Can you explain the concept of "bagging" in the context of Random Forest? How does it help in reducing overfitting and improving generalization?
4. How might the number of decision trees in a Random Forest affect the model's performance and training time? What is the trade-off associated with choosing a larger number of trees?
5. In this experiment, what are the possible safety levels that the Random Forest model predicts for cars? How can these predictions be useful for car manufacturers, consumers, and regulatory bodies?

4.7. Experiment No. 6

Aim: Reinforcement Learning

A) Implement Reinforcement Learning using an example of a maze environment that the agent needs to explore.

Objective:

To Study:

1. Develop an understanding of reinforcement learning concepts and algorithms.
2. Implement a maze environment that simulates the agent's exploration process.
3. Train an agent using reinforcement learning to learn optimal paths in the maze.

Theory:

Reinforcement Learning is a type of **machine learning** where an **agent** learns to make decisions by interacting with an **environment** to achieve a **goal**.

- The agent learns by **trial and error**.
- It receives **rewards or penalties** as feedback.
- The goal is to learn a **policy** that maximizes **cumulative reward** over time.

Unlike supervised learning (which learns from labeled data), RL **learns from the consequences of actions**, not from explicit input-output pairs.

Key Components of Reinforcement Learning:

Component	Description
Agent	The learner or decision-maker
Environment	The world with which the agent interacts
State (s)	A representation of the environment at a time
Action (a)	Choices made by the agent
Reward (r)	Feedback signal from environment (scalar value)
Policy (π)	Strategy used by the agent to choose actions based on states
Value Function (V)	Expected cumulative reward from a state
Model (optional)	The agent's representation of the environment dynamics

The RL Process (Interaction Loop) :

At each discrete time step ttt:

1. Agent observes current **state** sts_tst.
2. Agent chooses **action** ata_tat based on its policy π\piπ.
3. Environment responds with:
   - New **state** st+1s_{t+1}st+1
   - **Reward** rtr_trt
4. Agent updates its policy/knowledge to improve future decisions.

Formalizing Reinforcement Learning: Markov Decision Process (MDP):

Reinforcement Learning problems are commonly modeled as an MDP, defined by:

- A set of states S
- A set of actions A
- Transition probabilities $P(s'|s,a)$ — probability of moving to state s' after taking action a in state s
- Reward function $R(s,a,s')$ — reward received transitioning from s to s' via action a
- Discount factor $\gamma \in [0,1]$ — how future rewards are valued relative to immediate rewards

Objective: Maximize Expected Cumulative Reward:

The goal is to find a **policy** π that maximizes the expected **discounted cumulative reward**:

$$G_t - \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Common RL Algorithms

a) Value-Based Methods

- Learn value functions to derive policy.
- Example: **Q-Learning**, **SARSA**.

b) Policy-Based Methods

- Directly optimize the policy π(a|s).
- Example: **REINFORCE algorithm**.

c) Actor-Critic Methods

- Combine value and policy learning.
- Separate “actor” (policy) and “critic” (value).

Applications:

Reinforcement learning has numerous applications, including:

1. Game Playing: Training agents to play games like chess, Go, or video games.
2. Robotics: Teaching robots to perform tasks in real-world environments.
3. Autonomous Vehicles: Training self-driving cars to navigate safely on roads.
4. Recommendation Systems: Optimizing recommendations based on user interactions.
5. Resource Management: Finding optimal strategies in energy management or financial trading.

Input:

In this experiment, the inputs include:

1. Maze Environment: A representation of the maze, with walls, open paths, start, and goal locations.
2. Action Space: Possible actions the agent can take (e.g., move up, down, left, right).
3. Reward System: A defined reward structure, with positive and negative rewards.

Output:

1. Optimal Policy: The learned optimal actions for each state in the maze.
2. Agent Performance: The efficiency of the agent in reaching the goal and navigating the maze.

Conclusion:

Through this experiment, we successfully implemented reinforcement learning in a maze environment. The agent learned how to navigate the maze by exploring different paths and optimizing its decisions to maximize cumulative rewards. This experiment highlights the power of reinforcement learning in training agents to make sequential decisions in dynamic environments.

Outcome:

The outcome of this experiment is an agent that can efficiently explore the maze and reach the goal using learned optimal paths. This showcases the agent's ability to learn from interactions with its environment and make informed decisions to achieve its objective.