Empowerment Through Quality Technical Education

AJEENKYA DY PATIL SCHOOL OF ENGINEERING

Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohegaon, Pune – 412 105

Website: https://dypsoe.in/

LAB MANUAL

COMPUTER LABORATORY-I
MACHINE LEARNING (417521)

BE (AI&DS) 2020 COURSE

Course Coordinator

Prof. Priyanka Waghmare

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Department of Artificial Intelligence & Data Science

Vision:

Imparting quality education in the field of Artificial Intelligence and Data Science

Mission:

- To include the culture of R and D to meet the future challenges in AI and DS.
- To develop technical skills among students for building intelligent systems to solve problems.
- To develop entrepreneurship skills in various areas among the students.
- To include moral, social and ethical values to make students best citizens of country.

Program Educational Outcomes:

1. To prepare globally competent graduates having strong fundamentals, domain knowledge, updated with modern technology to provide the effective solutions for engineering problems.
2. To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.
3. To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.
4. To prepare the graduates with strong managerial and communication skills to work effectively as individuals as well as in teams.

Program Specific Outcomes:

1. Professional Skills- The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.

2. Problem-Solving Skills- The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

3. Successful Career and Entrepreneurship- The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies.

Table of Contents

Contents

1. Guidelines to manual usage

This manual assumes that the facilitators are aware of collaborative learning methodologies.

This manual will provide a tool to facilitate the session on Digital Communication modules in collaborative learning environment.

The facilitator is expected to refer this manual before the session.

Icon of Graduate Attributes

K Applying Knowledge	**A** Problem Analysis	**D** Design & Development	**I** Investigation of problems
M Modern Tool Usage	**E** Engineer & Society	**E** Environment Sustainability	**T** Ethics
T Individual & Team work	**O** Communication	**M** Project Management & Finance	**I** Life-Long Learning

Disk Approach- Digital Blooms Taxonomy

1: Remembering / Knowledge

2: Comprehension / Understanding

3: Applying

4: Analyzing

5: Evaluating

6: Creating / Design

Program Outcomes:

1. **Engineering knowledge:** An ability to apply knowledge of mathematics, including discrete mathematics, statistics, science, computer science and engineering fundamentals to model the software application.
2. **Problem analysis:** An ability to design and conduct an experiment as well as interpret data, analyze complex algorithms, to produce meaningful conclusions and recommendations.
3. **Design/development of solutions**: An ability to design and development of software system, component, or process to meet desired needs, within realistic constraints such as economic, environmental, social, political, health & safety, manufacturability, and sustainability.
4. **Conduct investigations of complex problems**:An ability to use research based knowledge including analysis, design and development of algorithms for the solution of complex problems interpretation of data and synthesis of information to provide valid conclusion.
5. **Modern tool usage:** An ability to adapt current technologies and use modern IT tools, to design, formulate, implement and evaluate computer based system, process, by considering the computing needs, limits and constraints.
6. **The engineer and society:** An ability of reasoning about the contextual knowledge of the societal, health, safety, legal and cultural issues, consequent responsibilities relevant to IT practices.
7. **Environment and sustainability:** An ability to understand the impact of engineering solutions in a societal context and demonstrate knowledge of and the need for sustainable development.
8. **Ethics:** An ability to understand and commit to professional ethics and responsibilities and norms of IT practice.
9. **Individual and team work :**An ability to apply managerial skills by working effectively as an individual, as a member of a team, or as a leader of a team in multidisciplinary projects.
10. **Communication:** An ability to communicate effectively technical information in speech, presentation, and in written form
11. **Project management and finance:** An ability to apply the knowledge of Information Technology and management principles and techniques to estimate time and resources needed to complete engineering project.
12. **Life-long learning:** An ability to recognize the need for, and have the ability to engage in independent and life-long learning.

Course Name: Computer Laboratory I – Machine Learning

Course Code: 417521

Course Outcomes

CO1: Implement regression, classification and clustering models
CO2: Integrate multiple machine learning algorithms in the form of ensemble learning
CO3: Apply reinforcement learning and its algorithms for real world applications
CO4: Analyze the characteristics, requirements of data and select an appropriate data model
CO5: Apply data analysis and visualization techniques in the field of exploratory data science
CO6: Evaluate time series data

CO to PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	2	3	-	-	-	2	2	1	1
CO2	3	3	3	2	3	-	-	-	2	2	1	1
CO3	3	3	3	2	3	-	-	-	2	2	1	1
CO4	3	2	2	3	3	-	-	-	2	1	1	1
CO5	3	2	2	3	3	-	-	-	2	1	1	1
CO6	3	2	2	3	3	-	-	-	2	2	1	1

CO to PSO Mapping:

	PSO1	PSO2	PSO3
CO1	2	1	1
CO2	2	2	1
CO3		1	1
CO4		2	1
CO5		1	1
CO6		2	1

2. Laboratory Objective

- Apply regression, classification and clustering algorithms for creation of ML models
- Introduce and integrate models in the form of advanced ensembles
- Conceptualized representation of Data objects
- Create associations between different data objects, and the rules
- Organized data description, data semantics, and consistency constraints of data

3. Laboratory Equipment/Software

Hardware:

1. PC or Workstation: A computer with sufficient processing power, RAM, and storage to handle data analysis tasks.

Software:

1. Python: A programming language commonly used for data analysis, machine learning, and visualization.
2. Jupyter Notebooks: An interactive computing environment for creating and sharing documents that contain live code, equations, visualizations, and narrative text.
3. Integrated Development Environment (IDE): Choose a Python IDE like PyCharm, VSCode, or JupyterLab for coding efficiency.
4. Libraries and Packages: Install and use the following Python libraries/packages:
    - NumPy: For numerical operations.
    - Pandas: For data manipulation and analysis.
    - Matplotlib: For basic 2D plotting.
    - Seaborn: For statistical data visualization.
    - Scikit-learn: For machine learning algorithms.
    - TensorFlow or PyTorch: For deep learning tasks.
    - Statsmodels: For statistical modelling.
    - Requests: For working with APIs.

4. Laboratory Experiment list

Sr. No	Title
	Prerequisite practical assignments or installation (if any)
1	Setting Up Python Environment
2	GitHub Repository Setup
	List of Assignments
1	Feature Transformation: 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
2	Regression Analysis: Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks: 1. Pre-process thedataset. 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
3	Classification Analysis: Implement K-Nearest Neighbours' algorithm on Social network ad dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset. Dataset link:https://www.kaggle.com/datasets/ rakeshrau/social-network-ads
4	Clustering Analysis: Implement K-Means clustering on Iris.csv dataset. Determine the number of clusters using the elbow method. Dataset Link: https://www.kaggle.com/datasets/uciml/iris.
5	Ensemble Learning: Implement Random Forest Classifier model to predict the safety of the car. Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set.
6	Reinforcement Learning: Solve the Taxi problem using reinforcement learning where the agent acts as a taxi driver to pick up a passenger at one location and then drop the passenger off at their destination.
	Content Beyond Syllabus
1	

4.1. Prerequisite

Setting Up Python Environment for Data Analysis and Machine Learning

Objective: The objective of this laboratory session is to guide you through the process of setting up a Python environment for data analysis and machine learning using the Anaconda distribution. By the end of this session, you should have a working Python environment with essential libraries installed, including NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, TensorFlow/PyTorch, and Jupyter Notebooks.

Task 1: Install Anaconda Distribution

1. **Download Anaconda:**
    - Go to the official Anaconda website: Anaconda Distribution.
    - Choose the appropriate version (Python 3.x recommended) for your operating system (Windows/Mac/Linux) and download the installer.
2. **Install Anaconda:**
    - Follow the installation instructions for your operating system.
    - During the installation, ensure you check the box that adds Anaconda to your system PATH.
3. **Verify Installation:**
    - Open a new terminal or command prompt.
    - Type **conda --version** and **python --version** to verify that Anaconda and Python have been successfully installed.

Task 2: Create a Virtual Environment

1. **Open a Terminal/Command Prompt:**
    - Open a new terminal or command prompt on your machine.
2. **Create a Virtual Environment:**
    - Run the following command to create a virtual environment named "env" (you can choose a different name):

```
(base) PS C:\Users\kusha> conda create --name env python=3.9.6
```

3. **Activate the Virtual Environment:**
    - Activate the virtual environment using:
        - On Windows: "**conda activate env"**
        - On MacOS/Linux: "**source activate env"**
4. **Verify Activation:**
    - The command prompt/terminal should now display the active virtual environment's name.

Task 3: Install Required Libraries

1. **Install Essential Libraries:** While in the virtual environment, install the necessary libraries using the following command:

```
(base) PS C:\Users\kusha> conda install numpy pandas matplotlib seaborn scikit-learn
```

2. **Install Deep Learning Libraries (Choose one):**
    - For TensorFlow:

```
(base) PS C:\Users\kusha> conda install tensorflow
```

3. **Install Jupyter Notebooks:**
    - Install Jupyter Notebooks for interactive coding and documentation:

```
PS C:\Users\kusha> conda install jupyter
```

4. **Verify Installations:**

    Open a Jupyter Notebook using the command **jupyter notebook** and check if you can import libraries without errors.

Conclusion: We have successfully set up a Python environment for data analysis and machine learning. This environment is now ready for use in various data science projects and assignments.

Laboratory Manual: GitHub Repository Setup for Version Control and Collaboration

Objective: The objective of this laboratory session is to guide you through the process of setting up a GitHub repository for version control and collaboration on code and assignments. By the end of this session, you should have a GitHub account, a new repository created for your assignments, and an initial Jupyter Notebook committed and pushed to the repository.

Task 1: Create a GitHub Account

1. **Visit GitHub:**
    - Go to the official GitHub website: GitHub.
    - If you already have an account, proceed to the next task. Otherwise, sign up for a new

Task 2: Create a New Repository

2. **Log In to GitHub:**
    - Log in to your GitHub account.
3. **Create a New Repository:**
    - Click on the "+" sign in the top right corner and select "New repository."
    - Fill in the repository name, add a description, choose public or private (based on your preference), and initialize it with a README file.
    - Click "Create repository."
4. **Copy Repository URL:**
    - After creating the repository, copy the repository URL from the "Code" button**.**

Task 3: Clone the Repository

1. **Open Terminal/Command Prompt:**
    - **Open a new terminal or command prompt on your local machine.**
2. **Navigate to Working Directory:**
    - **Navigate to the directory where you want to store your local copy of the repository.**

3. **Clone the Repository:**

    a. **Run the following command to clone the repository to your local machine:**

4. **Navigate into the Repository:**
    - **Change into the repository directory:**

Task 4: Initialize Jupyter Notebooks

1. Open Jupyter Notebooks:
    - Open a Jupyter Notebook in the cloned repository using the following command:
2. Create a New Jupyter Notebook:
    - In the Jupyter interface, create a new notebook for documenting and presenting your code.
3. Save the Notebook:
    - Save the notebook in the repository directory.

Task 5: Commit and Push

1. Add Changes:
    - In the terminal, add the changes you made to the local repository:
2. Commit Changes:
    - Commit the changes with a descriptive message:
3. Push to GitHub:
    - Push the changes to the GitHub repository:
    - Replace main with the branch name if it's different.

Conclusion: We have successfully set up a GitHub repository, cloned it to your local machine, and initialized a Jupyter Notebook. This repository is now ready for version control, collaboration, and documentation of your data science projects.

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:

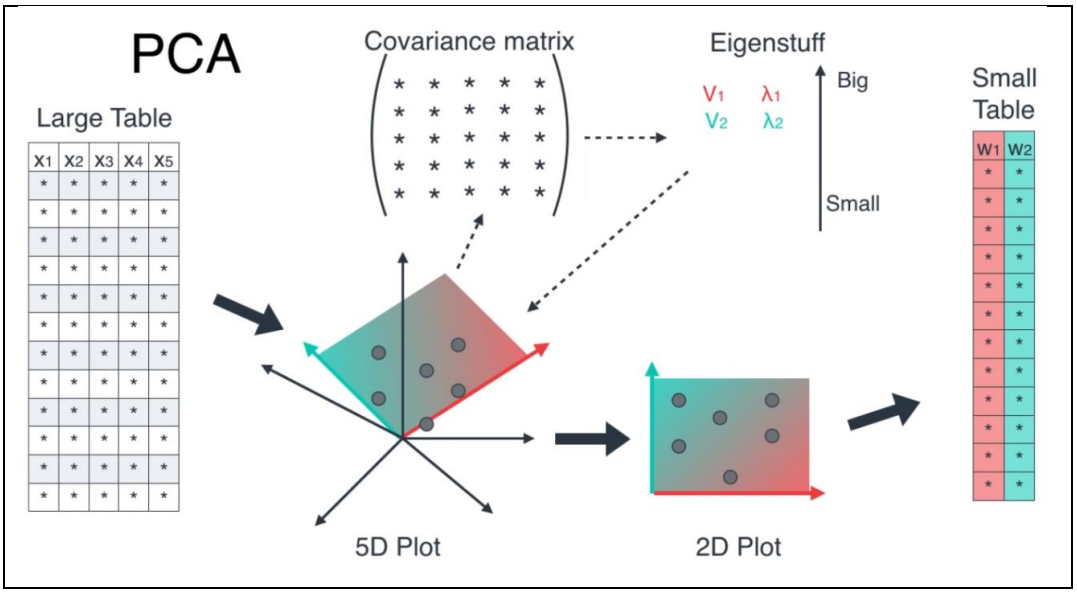
The main objectives of this experiment are to:

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:

Introduction to PCA:

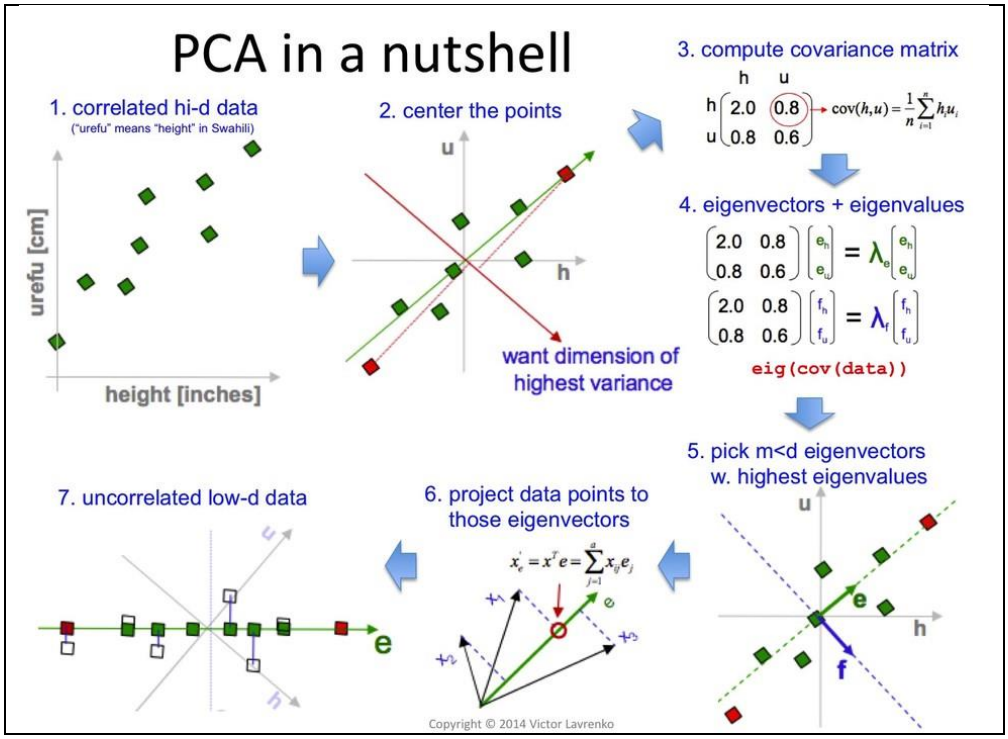
Principal Component Analysis (PCA) is a dimensionality reduction technique used to reduce the number of variables (features) in a dataset while retaining most of the original variance (information). PCA is particularly useful in datasets with a large number of interrelated variables, as it transforms the original features into a smaller set of uncorrelated components called **principal components**. These components capture the maximum variance in the data, helping simplify and visualize complex datasets.



2. Steps in PCA:

Let's break down how PCA works:

1. **Standardization:** Since PCA is affected by the scale of the features, the first step is to standardize the data by making all the variables have a mean of 0 and a standard deviation of 1.
2. **Covariance Matrix Computation:** The next step is to compute the covariance matrix for the standardized data, which tells how variables vary together.
3. **Compute Eigenvalues and Eigenvectors:** Eigenvalues explain the magnitude of the variance captured by each principal component, while eigenvectors define the direction of the new principal components.
4. **Select Principal Components:** Choose the top principal components that capture the most variance. These components are a linear combination of the original features.
5. **Transformation:** Transform the original dataset into the new space defined by the selected principal components.



Example:

Suppose you have a dataset with two variables, **X1** (age) and **X2** (income). PCA would transform this dataset by:

- Finding the directions (principal components) along which the data has the most variance.
- Reducing the data to one or two principal components (instead of the original variables).
- The result is a new dataset where most of the variance is captured by the principal components.

In a 2D plot, this would look like rotating the axes, so that the new axes (principal components) align with the directions of maximum variance.

Applications:

PCA has applications in various domains:

1. Image Compression: Reducing image dimensions while preserving key features.
2. Data Visualization: Visualizing high-dimensional data in a reduced space.
3. Feature Selection: Identifying important features for machine learning models.
4. Genomics: Analyzing gene expression data to discover hidden patterns.
5. Face Recognition: Representing faces with a smaller set of features.

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?

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:

Introduction to Regression Analysis:

Regression analysis is a statistical method used to model the relationship between a dependent variable (target) and one or more independent variables (features). The goal is to predict the value of the dependent variable based on the values of the independent variables.

Linear Regression: This is the most basic form of regression. It assumes a linear relationship between the independent variables and the dependent variable.

Ridge Regression: A type of linear regression that includes a regularization term to reduce overfitting by penalizing large coefficients.

Lasso Regression: Similar to ridge regression, but Lasso can set some coefficients to zero, effectively performing feature selection.

Steps in Regression Analysis:

Preprocess the Dataset: Handle missing data, categorical variables, and standardize/normalize data if needed.

Identify Outliers: Outliers can significantly impact the performance of regression models.

Check Correlation: Understanding feature relationships helps in deciding which features to keep in the model.

Apply Regression Models: Implement different regression models to predict the target variable.

Evaluate Models: Use metrics like R-squared (R^2), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) to evaluate model performance.

Uber Ride Price Prediction: Regression Analysis
We will perform regression analysis on the "Uber Fares Dataset" to predict the ride fare. Here's how to implement the workflow:

Step-by-Step Implementation:

Step 1: Preprocess the Dataset
Load the dataset and clean any missing or inconsistent data.
Convert categorical variables, such as pickup and drop-off locations, into numerical formats if needed.

Step 2: Identify Outliers
Detect and remove outliers using techniques such as the IQR method or Z-scores.

Step 3: Check Correlation
Assess the correlation between variables using a heatmap to find highly correlated features.

Step 4: Evaluate the Models
Compare model performance using metrics like R^2 and RMSE.

Explanation of Regression Analysis

Linear Regression:

- In linear regression, the relationship between the target variable and features is modeled as a straight line. For example, predicting Uber ride fare based on distance can be modeled as:

$$\text{Target} = \beta_0 + \beta_1 \times \text{Feature} + \epsilon$$

Ridge Regression:

- Ridge regression adds a regularization term (penalty) to the linear regression equation to reduce overfitting. The equation becomes:

$$\text{Cost Function} = \text{MSE} + \lambda \sum_{i=1}^{n} \beta_i^2$$

Lasso Regression:

- Lasso regression also includes a regularization term but can shrink some coefficients to zero, effectively selecting features:

$$\text{Cost Function (Lasso)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

1. Linear Regression Example

Scenario:

You are trying to predict the fare of an Uber ride based on the distance of the trip.

Formula:

Fare=β0+β1×Distance\text{Fare} = \beta_0 + \beta_1 \times \text{Distance}Fare=β0+β1×Distance

Example:

- **Distance (miles)**: 2, 5, 10, 12
- **Fare (in dollars)**: 6, 15, 30, 36

Using linear regression, you estimate that:

$$\text{Fare} = 2 + 3 \times \text{Distance}$$

- The intercept β0=2\beta_0 = 2β0=2 suggests that even if the distance is 0, the fare starts at $2 (a base fare).
- The slope β1=3\beta_1 = 3β1=3 tells us that the fare increases by $3 for every additional mile traveled.

Here:

- β0=2\beta_0 = 2β0=2 (intercept): This represents a base fare of $2 when the distance is 0 miles.
- β1=3\beta_1 = 3β1=3 (slope): This means the fare increases by $3 for every additional mile.

If a ride is 8 miles, the predicted fare is:

$$\text{Fare} = 2 + 3 \times 8 = 26 \text{ dollars}$$

Example 2: Lasso Regression – Predicting Uber Fare

You want to predict the fare of an Uber ride based on the distance traveled and several other features. You decide to use Lasso Regression to help select the most important features while shrinking less important ones to zero.
Linear Regression Formula:
The model can be expressed as:

Data:

You have the following dataset:

Distance (miles)	Traffic (scale 1-5)	Weather (1=Bad, 0=Good)	Fare (dollars)
2	3	0	6
5	4	1	15
10	2	0	30
12	1	1	36

Estimation Using Lasso Regression:

After fitting a Lasso regression model, you might get the following coefficients:

- $\beta_0 = 2$
- $\beta_1 = 3$ (for Distance)
- $\beta_2 = 0$ (for Traffic, shrunk to zero)
- $\beta_3 = 1$ (for Weather)

This results in the model:

$$\text{Fare} = 2 + 3 \times \text{Distance} + 0 \times \text{Traffic} + 1 \times \text{Weather}$$

Prediction:

To predict the fare for an 8-mile ride with good weather:

$$\text{Fare} = 2 + 3 \times 8 + 1 \times 0 = 26 \text{ dollars}$$

Example 2: Ridge Regression – Predicting Uber Fare

You are again predicting the fare of an Uber ride based on similar features but will use Ridge Regression this time to avoid overfitting and keep all features in the model.

Data:

The same dataset is used:

Distance (miles)	Traffic (scale 1-5)	Weather (1=Bad, 0=Good)	Fare (dollars)
2	3	0	6
5	4	1	15
10	2	0	30
12	1	1	36

Estimation Using Ridge Regression:

After fitting a Ridge regression model, you might get the following coefficients:

- $\beta_0 = 1$
- $\beta_1 = 2.5$ (for Distance)
- $\beta_2 = 1.5$ (for Traffic)
- $\beta_3 = 1$ (for Weather)

This results in the model:

$$\text{Fare} = 1 + 2.5 \times \text{Distance} + 1.5 \times \text{Traffic} + 1 \times \text{Weather}$$

Prediction:

To predict the fare for an 8-mile ride with bad weather and traffic at level 2:

$$\text{Fare} = 1 + 2.5 \times 8 + 1.5 \times 2 + 1 \times 1 = 1 + 20 + 3 + 1 = 25 \text{ dollars}$$

Applications:

Regression analysis has applications in various domains:

1. Economics: Predicting economic indicators based on various factors.
2. Real Estate: Estimating house prices based on features like area and location.
3. Finance: Predicting stock prices or interest rates.
4. Healthcare: Predicting patient outcomes based on medical parameters.
5. Marketing: Forecasting sales based on advertising expenditures.

Input:

The input for this experiment is the Uber ride price dataset, which contains features related topickup 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?

Experiment No. 3

Aim: Classification Analysis:

A) Implement K-Nearest Neighbours' algorithm on Social network ad dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset. Dataset link:https://www.kaggle.com/datasets/rakeshrau/social-network-ads

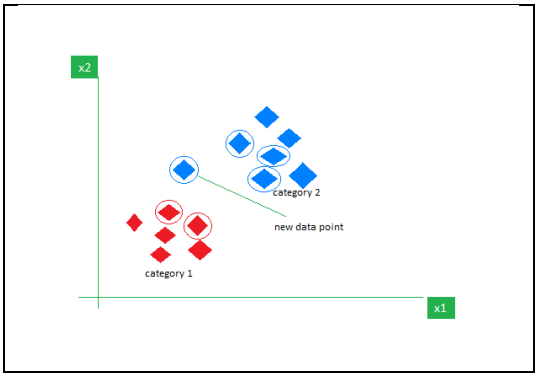
Objective:

1. Implement a K-Nearest Neighbours' algorithm on Social network ad dataset
2. Train the model on a dataset of labeled images to learn patterns and characteristics of different digits.
3. Evaluate the performance of the trained KNN model in terms of classification accuracy.

Theory:

1. Introduction to KNN:

K-Nearest Neighbors (KNN) is a simple, non-parametric, and lazy learning algorithm used for classification and regression tasks. In KNN classification, the algorithm classifies a data point based on how its neighbors are classified. The "K" in KNN represents the number of nearest neighbors considered for the decision-making process.



Theory of KNN with Example:

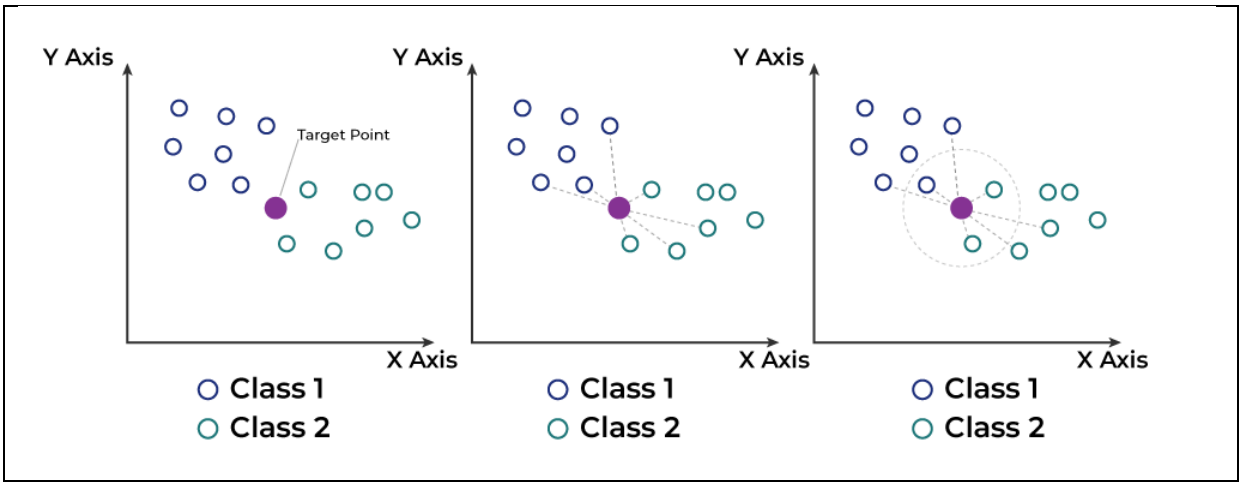
In KNN, an unlabeled data point is classified by considering the majority label among its "K" nearest data points.

Example:

Assume you have a dataset of social network users with features such as **age** and **salary** and a binary target variable indicating whether they purchased a product (1 for "Yes," 0 for "No"). You want to predict whether a new user will purchase a product based on their age and salary.

Steps:

1. **Calculate Distance:** Compute the distance between the new user's data point and all other data points in the dataset (using Euclidean or another distance metric).
2. **Find K Neighbors:** Identify the K closest data points to the new user.
3. **Majority Vote:** The class (0 or 1) with the majority of votes among the K neighbors is assigned to the new user.



2. KNN Implementation on Social Network Ads Dataset

Here's how to implement the KNN algorithm using the Social Network Ads dataset:

- **Confusion Matrix:** The confusion matrix will show the number of correct and incorrect predictions in each category (True Positives, False Positives, True Negatives, False Negatives).

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

True Positives (TP): Cases where the model correctly predicted the positive class.
True Negatives (TN): Cases where the model correctly predicted the negative class.
False Positives (FP): Cases where the model incorrectly predicted the positive class (Type I error).
False Negatives (FN): Cases where the model incorrectly predicted the negative class (Type II error).

- **Accuracy:** The proportion of correct predictions made by the model. Accuracy is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** The ratio of true positive predictions to the total positive predictions (i.e., how many of the predicted positives were actually positive)

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** The ratio of true positive predictions to the total actual positives (i.e., how many actual positives were predicted correctly)

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Error Rate:** The proportion of incorrect predictions made by the model:

$$\text{Error Rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

$$\text{Error Rate} = 1 - \text{Accuracy}$$

Applications:

1. **Recommendation Systems**: Suggest products or content based on user preferences by finding similar users or items.
2. **Image Recognition**: Classify images by comparing them with a database of labeled images, identifying the closest matches.
3. **Medical Diagnosis**: Predict diseases by comparing patient data with historical cases, aiding in early detection and treatment.
4. **Finance**: Detect fraudulent transactions by comparing them with known cases of fraud, enhancing security measures.

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 KNN model capable of classifying image recognition.

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 KNNs in image classification tasks.

Outcome:

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

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:

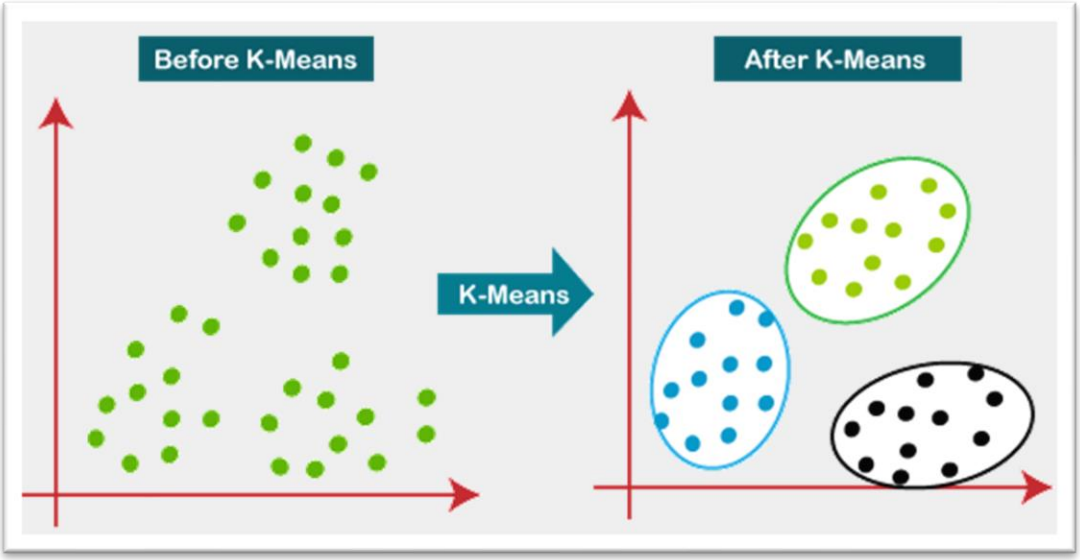
The main objective of this experiment is to understand and implement K-Means clustering and the elbow method for selecting the appropriate number of clusters in the given Iris dataset.

Theory:

K-Means clustering is a popular unsupervised machine learning algorithm that partitions a dataset into distinct clusters based on the similarity between data points. The objective of K-Means is to group data points such that points within the same cluster are as close as possible, while clusters themselves are as distinct as possible from each other.

How K-Means Works:

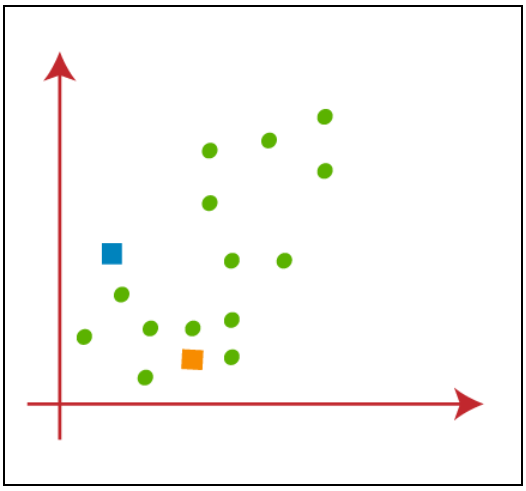
1. **Initialize Cluster Centers (Centroids):** Choose `K` random points from the dataset to serve as initial cluster centers.
2. **Assign Points to Nearest Centroid:** Each data point is assigned to the cluster whose centroid is closest to it. This is done based on a distance metric, usually Euclidean distance.
3. **Update Centroids:** Once all points are assigned to clusters, the centroids are updated by taking the mean of all the points in each cluster.
4. **Repeat:** Steps 2 and 3 are repeated iteratively until the centroids no longer change significantly, or a specified number of iterations is reached.



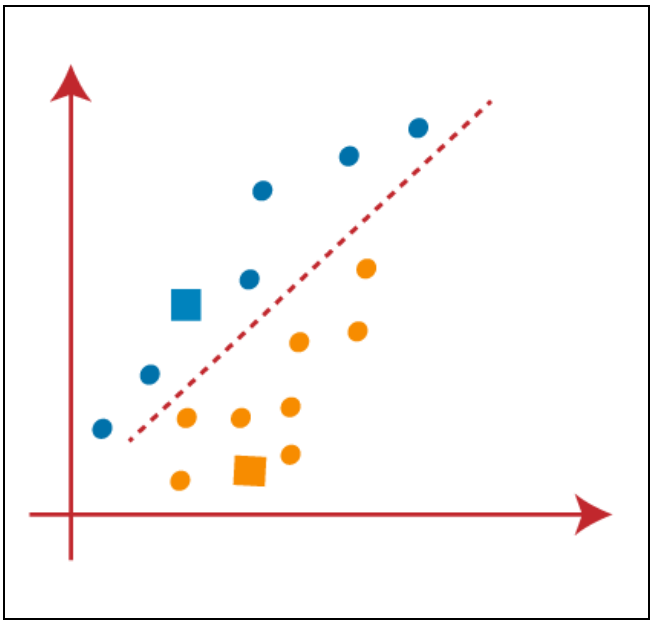
Example:

Imagine a dataset with customer shopping habits where each data point represents a customer with two features: annual income and spending score. The goal is to segment the customers into clusters based on these two features:

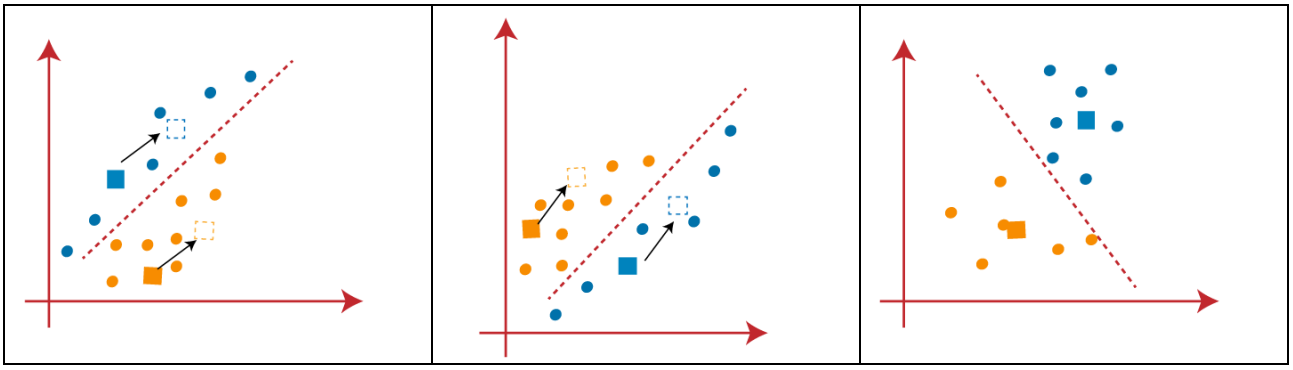
1. If you set `K=2`, the K-Means algorithm will randomly initialize three centroids.



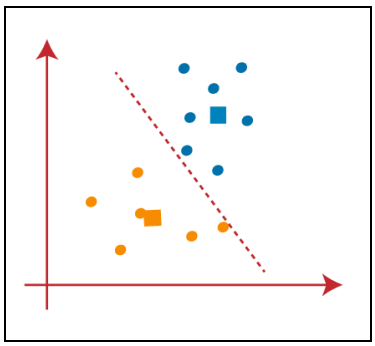
2. Customers are assigned to one of these three clusters based on which centroid is closest.



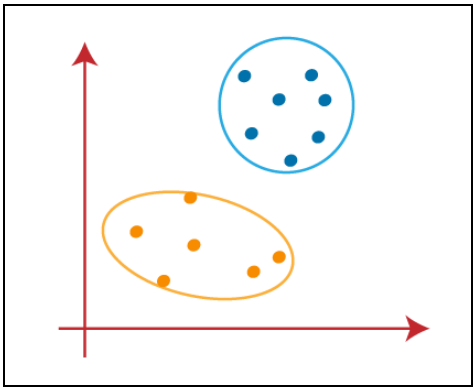
3. The centroids are updated based on the average income and spending score of customers in each cluster.



4. This process continues until the centroids stabilize.



The final clusters represent distinct groups of customers with similar shopping behavior.

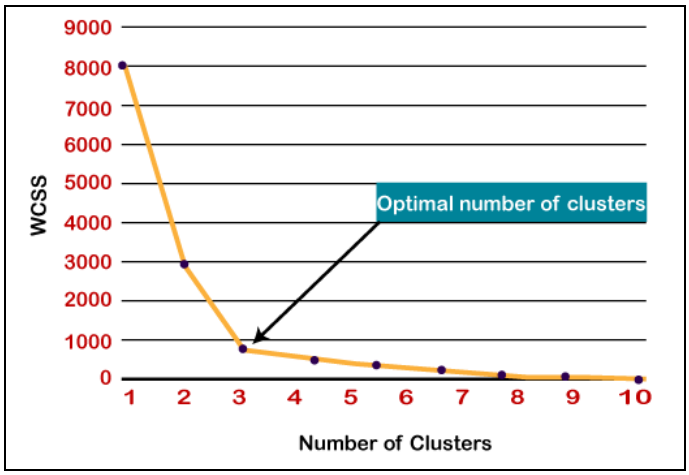


Elbow Method for Determining the Number of Clusters

The elbow method is a heuristic used to determine the optimal number of clusters K in K-Means clustering. It helps find the point where adding more clusters doesn't significantly reduce the within-cluster sum of squares (WCSS), which measures the variance within clusters.

Steps in the Elbow Method:

1. **Run K-Means for a range of K values (e.g., 1 to 10).**
2. **Calculate the WCSS** for each value of K. WCSS is the sum of squared distances between each point and its corresponding centroid.
3. **Plot K vs. WCSS** on a graph.
4. The optimal number of clusters is the "elbow" point where the rate of decrease in WCSS slows down significantly. This is the point where adding more clusters provides diminishing returns in reducing WCSS.



In other words, the elbow point is where the clustering model performs well without overfitting the data.

Example of the Elbow Method

1. **Generate Data**: Imagine you have a dataset with 300 points that are grouped into 5 clusters. Each cluster has its own center, and the points are spread around these centers.
2. **Choose a Range for k**: Decide on a range of cluster numbers (k) to test. For instance, you might test from 1 to 10 clusters.
3. **Fit K-Means for Each k**:
   - For each value of k, apply the K-Means clustering algorithm to the dataset.
   - The algorithm will assign each data point to one of the k clusters.
4. **Calculate Inertia**: After clustering for each k, calculate the inertia (or within-cluster sum of squares), which measures how tightly the points in each cluster are packed around the cluster center. Lower inertia means the points are closer to their respective cluster centers.

5. **Record Inertia Values**: Create a list to store the inertia values corresponding to each k value you tested.
6. **Plot the Results**:
    - Create a plot with the number of clusters (k) on the x-axis and the inertia values on the y-axis.
    - Each point on the graph corresponds to a k value and its associated inertia.
7. **Identify the Elbow Point**: Look at the plot and find the point where the inertia starts to decrease at a slower rate (this is the "elbow"). This point indicates the optimal number of clusters.

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?

Experiment No. 5

Aim: Ensemble Learning:

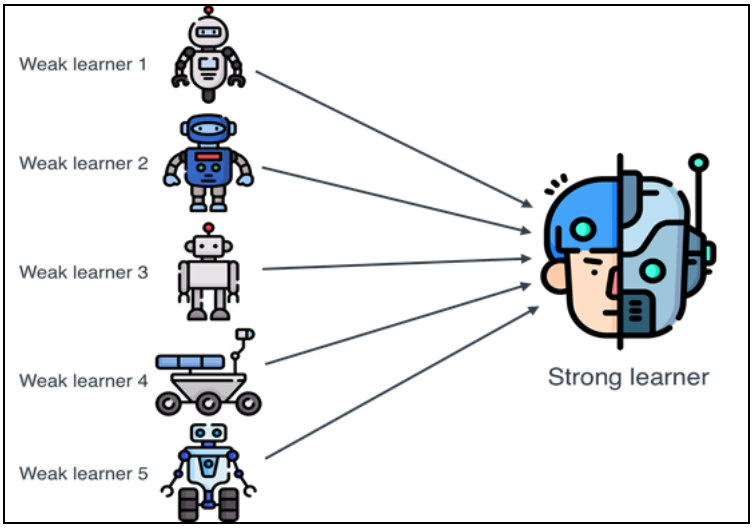
A) Implement Random Forest Classifier model to predict the safety of the car. Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set

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:

Ensemble learning is a machine learning technique where multiple models (often referred to as "weak learners") are combined to form a stronger, more accurate model. The idea behind ensemble methods is that by aggregating the predictions from multiple models, the overall model can achieve higher accuracy and better generalization than any individual model.

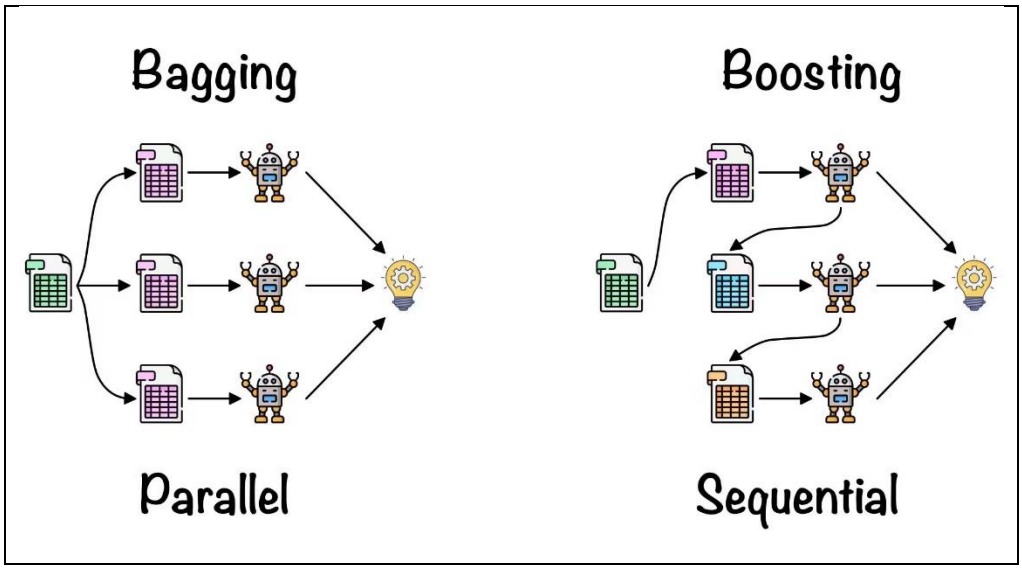


Ensemble learning can help reduce overfitting, improve predictive performance, and make models more robust by reducing the variance and bias in the predictions.

There are two main types of ensemble methods:

1. **Bagging (Bootstrap Aggregating):**
   - The key idea is to create multiple subsets of the dataset by sampling with replacement (bootstrapping) and then train a model (e.g., decision trees) on each subset.
   - The final prediction is made by averaging the predictions (for regression) or taking a majority vote (for classification) from all the models.
   - **Example:** Random Forest is a bagging-based algorithm.

2. **Boosting**
    - Boosting works by training models sequentially, where each new model tries to correct the mistakes made by the previous one. It places more focus on incorrectly predicted instances.
    - The final model is a weighted combination of all the weak learners.
    - **Example:** AdaBoost, Gradient Boosting.



Example of Ensemble Learning:

Let's consider a classification problem where we want to predict whether an email is spam or not (binary classification). Instead of relying on a single decision tree, we can use an ensemble of multiple decision trees:

1. **Bagging Example:** A Random Forest classifier trains multiple decision trees on different random subsets of the email dataset. Each tree makes a prediction (spam/not spam), and the final prediction is based on the majority vote across all trees. This helps to reduce overfitting because individual decision trees may vary due to randomness, but the overall model is more stable.
2. **Boosting Example:** In AdaBoost, the algorithm first trains a weak classifier (such as a simple decision stump) on the email data. Then, it adjusts the weights of misclassified emails and trains another classifier, focusing more on the misclassified data. This process is repeated multiple times, and the final classifier is a combination of all weak classifiers with different weights.

Random Forest Classifier:

Random Forest is a bagging-based ensemble learning algorithm that creates a collection of decision trees, each trained on a random subset of the training data. It is particularly effective for classification tasks like predicting car safety because:

- It reduces overfitting compared to a single decision tree.
- It handles high-dimensional data well.

- It can deal with missing or noisy data effectively.

Steps for Random Forest Classifier:

1. **Data Bootstrapping:** Create several random subsets of the training data using bootstrapping.
2. **Model Training:** Train a decision tree on each subset.
3. **Random Feature Selection:** At each split in the decision tree, select a random subset of features to determine the best split. This further introduces randomness and reduces correlation between trees.
4. **Prediction Aggregation:** For classification, the final prediction is made by taking a majority vote from all the individual trees.

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?

Experiment No. 6

Aim:

Reinforcement Learning: Solve the Taxi problem using reinforcement learning where the agent acts as a taxi driver to pick up a passenger at one location and then drop the passenger offat their destination.

Objective:

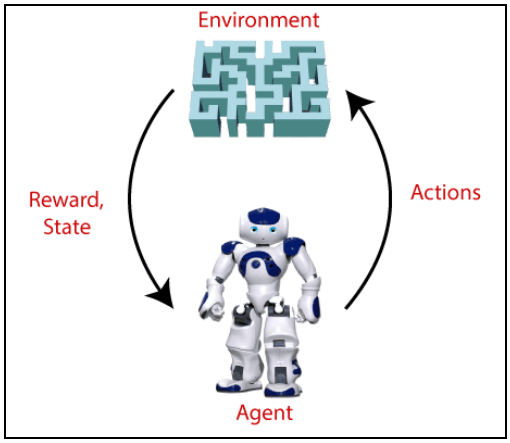
The objective of this experiment is to study and implement reinforcement learning (RL) by solving the classic Taxi problem, where the agent acts as a taxi driver to pick up and drop off passengers efficiently using learned strategies.

Theory:

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with its environment. The agent takes actions based on the state of the environment, and it receives feedback in the form of rewards or penalties. The goal is to learn a policy that maximizes the cumulative reward over time by exploring the environment and learning from the consequences of actions.

Key Concepts in Reinforcement Learning:

1. **Agent:** The learner or decision-maker (e.g., the taxi).
2. **Environment:** The world in which the agent operates (e.g., the taxi grid with locations and passengers).
3. **State (S):** A representation of the current situation of the agent within the environment (e.g., the taxi's location, passenger location, and destination).
4. **Action (A):** The choices available to the agent to affect the environment (e.g., move north, south, east, or west; pick up or drop off a passenger).
5. **Reward (R):** A feedback signal that tells the agent how good or bad its actions were in a given state (e.g., +20 for successfully dropping off a passenger, -10 for illegal moves).
6. **Policy (π):** A strategy that the agent uses to decide what actions to take in different states.
7. **Value Function (V):** Measures the long-term expected reward from being in a state and following a certain policy.



Reinforcement Learning Process:

1. **Exploration vs. Exploitation:** The agent needs to balance exploration (trying new actions to discover better strategies) and exploitation (using known actions to maximize reward). Techniques like ε-greedy are used to balance this tradeoff.
2. **Q-Learning:** One of the most common RL algorithms. It is an off-policy method where the agent learns a value function, called the Q-function, that represents the expected cumulative reward for taking a particular action in a specific state and following the optimal policy thereafter.

   Q-value Update Formula:

   Q(s,a)=Q(s,a)+α(r+γmax⁡a′Q(s′,a′)−Q(s,a))Q(s, a) = Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)Q(s,a)=Q(s,a)+α(r+γa′maxQ(s′,a′)−Q(s,a))

   Where:

   - `s` is the current state
   - `a` is the action taken
   - `r` is the reward received
   - `s'` is the new state after taking action `a`
   - α is the learning rate (controls how much new information overrides old information)
   - γ is the discount factor (controls how much future rewards are considered)

   The goal is to update the Q-values to reflect the best possible action-value pairs in the long run.

Example: Taxi Problem

The **Taxi problem** is a classic example in reinforcement learning. In this problem, the agent (the taxi) must:

- Pick up a passenger from a specific location.
- Drop off the passenger at the desired destination.
- Navigate through a grid-based environment, while avoiding obstacles and minimizing time and penalties.

Environment Setup:

- The environment is typically represented as a 5x5 grid, where:
  - The taxi can move in four directions (north, south, east, west).
  - The taxi can perform actions like "pick up" and "drop off."
  - There are fixed locations where passengers can be picked up and dropped off.

Reward System:

- The taxi receives a positive reward (+20) for successfully dropping off a passenger.

- A small negative reward (-1) is given for each movement step to encourage the taxi to find the quickest route.
- Larger negative rewards (-10) are given for illegal actions, such as trying to pick up or drop off a passenger in the wrong location.

Goal:

The agent must learn the optimal policy to navigate the grid, pick up the passenger, and drop them off as efficiently as possible while maximizing the cumulative reward.

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.

Questions:

1. What is the primary objective of implementing Reinforcement Learning in the context of the maze exploration experiment?
2. Explain the concept of Q-learning and its significance in training an agent to navigate a maze.
3. Can you provide an example of a real-world application where Reinforcement Learning, similar to the maze exploration scenario, could be used to solve a complex problem? Describe the problem and how RL could address it.
4. In the context of the maze environment, what components make up the agent's learning process? How do these components interact to improve the agent's decision-making abilities?
5. What are some factors that could affect the efficiency and speed at which the agent learns to navigate the maze? How might adjusting the reward structure influence the agent's behaviour and learning speed?