Faculty of Technology and Engineering

Department of Computer Science & Engineering

Laboratory Manual

Academic Year	:	2024-25	Semester	:	5th
Course code	:	CSE303	Course name	:	Machine Learning

Note: The laboratory will emphasize the use of Python, Python Packages, Machine Learning and itsapplications.

Instructions: 1All Practical must be performed with different data sets from the list of datasets given.

Sr. No	Index	Hrs.	CO	Bloom's Taxonomy	Pg. No
1	To understand and apply basic operations of Numpy, Pandas, and Matplotlib for data manipulation, analysis, and visualization.	4	2	Apply, Analyze, Evaluate	2
2	To understand and apply the linear regression algorithm for the prediction and evaluate its performance.	4	3		3
3	To understand and apply the logistic regression algorithm for binary classification	4	3		5
4	To understand and apply the KNN algorithm for multi-class classification to classify instances into multiple categories based on their features	4	3		6
5	The aim of this case study is to demonstrate the application of Principal Component Analysis (PCA) for feature selection and to compare the performance of Naïve Bayes and Support Vector Machine (SVM) classifiers using the full dataset and a reduced feature set.	4	3,4		7
6	To understand and implement K-means and DBSCAN clustering algorithms on a given dataset, compare their performances, and interpret the results	2	4		8
7	To understand and implement a Neural Network with Pima Indians Diabetes dataset using Keras.	2	3		10
8	To understand and implement the Q-Learning algorithm for solving a reinforcement learning problem and evaluate its performance.	4	5		11
9	The aim of this case study is to implement a Convolutional Neural Network (CNN) for the binary classification of images of dogs and cats. The goal is to build a model that can accurately distinguish between images of dogs and cats.	2	5		13

Annexure

Practical 1: Exploring Foundational Tools for Machine Learning

Aim:

To understand and apply basic operations of Numpy, Pandas, and Matplotlib for data manipulation, analysis, and visualization.

Objectives:

- Learn to create and manipulate arrays using Numpy.
- Perform data manipulation and analysis with Pandas DataFrames.
- Visualize data using Matplotlib charts and plots.
- Compare performance between Numpy and Python lists for large-scale matrix operations.

Description:

The Iris dataset is a well-known dataset in machine learning and statistics, containing information about three species of Iris flowers: Setosa, Versicolor, and Virginica. Each species has 50 samples, making a total of 150 samples in the dataset. For each flower sample, four features are measured: sepal length, sepal width, petal length, and petal width, all in centimeters. The target variable is the species of the Iris flower.

Dataset link:

archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Tasks:

Part 1: NumPy

1. **Array Creation:**
    - Create a NumPy array with zeros, ones, and random values.
    - Generate an array with a specific sequence of numbers (e.g., arithmetic progression).
2. **Slicing and Updating:**
    - Extract specific portions of an array using slicing syntax.
    - Modify elements within an array using indexing and assignment.
3. **Shape Manipulation:**
    - Reshape an array into different dimensions (e.g., from 1D to 2D).
    - Transpose an array to swap rows and columns.
4. **Looping:**
    - Iterate through an array using for loops and perform operations on elements.
5. **File Reading:**
    - Load data from a text file (CSV or similar) using NumPy functions (e.g., np.genfromtxt).
6. **Performance Benchmarking:**
    - Measure the execution time for matrix multiplication using NumPy arrays vs. Python lists for a 1000x1000 matrix. Analyze the performance difference.

Part 2: Pandas

1. **DataFrame Creation:**
    - Construct a DataFrame from scratch, specifying column names and data types.
2. **File Reading:**

- Read the chosen dataset from its source file format (CSV, Excel) into a Pandas DataFrame.
- Deal with missing values in the data

3. **Slicing and Manipulation:**
   - Select specific rows and columns using various indexing methods (e.g., by label, position).
   - Filter data based on specific conditions using boolean expressions.
4. **Data Export:**
   - Export the DataFrame to a new CSV file or another desired format.
5. **Loops and Row/Column Manipulation:**
   - Employ loops to perform calculations or modifications on rows or columns within the DataFrame.
6. **Masking:**
   - Apply boolean masks to filter and select data based on specific criteria.
   - Read data selectively based on a boolean DataFrame.

Part 3: Matplotlib

1. **Importing and Configuration:**
   - Import necessary modules from Matplotlib for plotting.
   - Configure plot elements like labels, titles, and grid lines.
2. **Line Charts:**
   - Create a line chart to visualize a numerical variable over time or another relevant category.
3. **Correlation Matrix:**
   - Generate a correlation matrix to depict relationships between features in the dataset.
4. **Histograms:**
   - Construct histograms to analyze the distribution of numerical features in the dataset.
   - Detect if Outliers exist and Plot the data distribution using Box Plots.
5. **Multivariate Data Visualization:**
   - Create scatter plots or other visualization techniques to represent multivariate data (multiple features).
6. **Pie Charts:**
   - Design a Pie chart to represent the proportions of categorical data within a specific feature.

Questions to be Answered:

1. How does the performance of Numpy matrix multiplication compare to the Python list implementation?
2. What are the advantages of using Numpy arrays over Python lists for large-scale data operations?
3. How can you efficiently iterate through large NumPy arrays using vectorized operations instead of loops?
4. How can you filter rows in a DataFrame based on a condition?
5. How can you handle missing values within a Pandas DataFrame?
6. What are the benefits of using DataFrames for data manipulation compared to Numpy arrays?
7. How can you visualize the correlation between two variables?
8. What types of plots are best suited for visualizing categorical vs. numerical data?
9. Explore ways to combine multiple plots or visualizations into a single figure using Matplotlib techniques.

Practical 2: Linear Regression

Aim:

To understand and apply the linear regression algorithm for the prediction and evaluate its performance.

Objectives:

- Gather and pre-process the housing dataset to prepare it for analysis.
- Implement the linear regression algorithm to train a predictive model.
- Evaluate the model's performance and interpret its results.
- Understand the impact of different features on house prices through analysis.

Description:

The Boston Housing Dataset contains information collected by the U.S. Census Service concerning housing in the area of Boston, Massachusetts. The dataset consists of 506 instances, each representing a different suburb of Boston. Each suburb is described by 14 attributes or features, including variables such as crime rate, proportion of residential land zoned for lots over 25,000 square feet, average number of rooms per dwelling, and others. The target variable is the median value of owner-occupied homes (in thousands of dollars). Apply linear regression algorithm for the following tasks and predict the house prices.

Dataset link:

https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset

Tasks:

1. **Data Preparation:**
    - Load and inspect the dataset.
    - Handle missing values and outliers if present.
    - Perform feature scaling or normalization if necessary.
2. **Model Training and Evaluation:**
    - Split the dataset into training and test sets.
    - Implement linear regression using a scikit-learn library.
    - Train the model on the training set and make predictions on the test set.
    - Evaluate the model using metrics such as mean squared error (MSE) and R-squared.
3. **Analysis and Interpretation:**
    - Analyze the coefficients of the linear regression model to understand the importance of different features.
    - Visualize the relationship between the predicted prices and actual prices.
    - Identify any trends or patterns in the residuals to assess model performance.

Questions to be Answered:

1. How does the linear regression model perform in predicting house prices based on the chosen features?
2. Which features have the most significant impact on house prices according to the model?
3. What are the strengths and limitations of using linear regression for predicting house prices in this context?
4. How could the model be improved or extended for better performance?

Practical 3: Logistic Regression

Aim:

To understand and apply the logistic regression algorithm for binary classification.

Objectives:

- Explore and understand the structure and characteristics of the dataset related to diabetes prediction.
- Build a logistic regression model to classify individuals into diabetic or non-diabetic categories.
- Evaluate the model's performance using appropriate metrics.
- Interpret the results and gain insights into the factors contributing to diabetes risk.

Description:

The Pima Indians Diabetes Database is a dataset collected by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). It comprises health data from Pima Indian women aged 21 and older, focusing on factors that may influence the development of diabetes. The dataset includes variables such as the number of pregnancies, glucose concentration, blood pressure, skin thickness, insulin levels, BMI, age, and a binary outcome indicating whether the individual developed diabetes within five years of the measurements. Apply logidtic regression algorithm for the following tasks and perform binary classification.

Dataset link:

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/data

Tasks:

1. **Data Exploration and Preprocessing:**
- Load and inspect the dataset to understand its structure and attributes.
- Handle missing values and perform data cleaning as necessary.
- Explore the distributions of variables and correlations using histograms, box plots, and correlation matrices.

2. **Feature Selection and Engineering:**
- Select relevant features that may influence the likelihood of diabetes based on domain knowledge and exploratory data analysis.
- Optionally, transform or normalize features to improve model performance.

3. **Model Building and Evaluation:**
- Split the dataset into training and test sets.
- Implement logistic regression using a scikit-learn library.
- Train the model on the training set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.

- Compare the performance of the logistic regression model with other classification algorithms if applicable.

4. **Interpretation and Insights:**

- Interpret the coefficients of the logistic regression model to understand the impact of each feature on the likelihood of diabetes.
- Visualize the model's predictions using ROC curves and precision-recall curves to assess its discriminative ability.
- Discuss insights gained regarding the factors contributing to diabetes risk and potential implications for preventive healthcare strategies.

Questions to be Answered:

1. How accurately can the logistic regression model predict the likelihood of diabetes based on the selected features?
2. Which features have the most significant impact on predicting diabetes risk according to the model?
3. What are the strengths and limitations of using logistic regression for diabetes prediction in this context?
4. How could the model be improved or extended for better performance, and what additional data or features might be beneficial?

Practical 4: K-Nearest Neighbour

Aim:

To understand and apply the KNN algorithm for multi-class classification to classify instances into multiple categories based on their features.

Objectives:

- Explore and understand the dataset structure and attributes suitable for multiclass classification.
- Implement the KNN algorithm to classify instances into one of several predefined classes.
- Evaluate the performance of the KNN model using appropriate metrics.
- Interpret the results and understand the strengths and limitations of the KNN approach for multiclass classification.

Description:

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of 70,000 grayscale images of handwritten digits (0-9), with 60,000 images in the training set and 10,000 images in the test set. Each image is a 28x28 pixel array, representing a single digit from 0 to 9. Apply KNN algorithm for the following tasks and perform the multi-class classification.

Dataset link:

https://www.kaggle.com/datasets/hojjatk/mnist-dataset

Tasks:

1. **Data Exploration and Pre-processing:**

- Load and visualize the dataset to understand its structure and features.

- Check for missing values and handle them if necessary.
- Explore the distributions of variables and relationships between features using histograms, scatter plots, and pair plots.

2. Model Implementation and Evaluation:

- Split the dataset into training and test sets.
- Implement the KNN algorithm using a scikit-learn library.
- Choose an appropriate value for the K parameter through cross-validation or other methods.
- Train the KNN model on the training set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score for each class.
- Visualize the results, such as confusion matrices or classification reports, to assess the model's classification accuracy and ability to generalize.

3. Interpretation and Insights:

- Interpret the model's predictions to understand how well it classifies instances into different classes.
- Analyze the impact of the choice of K on model performance and discuss any trade-offs between bias and variance.
- Discuss the limitations of the KNN algorithm for multiclass classification tasks, such as sensitivity to the choice of distance metric and computational efficiency.
- Compare the performance of KNN with other classification algorithms if applicable, and identify scenarios where KNN is particularly effective or ineffective.

Questions to be Answered:

1. Which machine learning algorithm(s) achieve the highest accuracy in classifying handwritten digits from the MNIST dataset?
2. How does the performance of deep learning model CNN compare to traditional machine learning algorithms for digit recognition?
3. What are the challenges and limitations of using the MNIST dataset for training and evaluating digit recognition models?
4. How can insights gained from this analysis be applied to other image classification tasks or real-world applications?

Practical 5: Feature Selection and Model Performance Analysis Using PCA

Aim:

The aim of this case study is to demonstrate the application of Principal Component Analysis (PCA) for feature selection and to compare the performance of Naïve Bayes and Support Vector Machine (SVM) classifiers using the full dataset and a reduced feature set.

Objectives

1. To understand the concept of feature selection using PCA.
2. To apply PCA for selecting the K best features from a dataset.
3. To train and evaluate the performance of Naïve Bayes and SVM classifiers.
4. To compare the accuracy and training time of the models with the full dataset and with the selected K features.

Description

In this case study, we will use a dataset where the number of samples is smaller than the number of features. The dataset will be sourced from the UCI Machine Learning Repository. The practical implementation will involve the following steps:

1. **Data Loading and Exploration**: Load and explore the dataset to understand its structure and characteristics.
2. **Data Preprocessing**: Perform any necessary data preprocessing steps such as handling missing values and standardizing the features.
3. **PCA for Feature Selection**: Apply PCA to the dataset to select the K best features.
4. **Model Training and Evaluation**: Train Naïve Bayes and SVM classifiers on the full dataset and on the dataset with selected K features. Evaluate the models' performance in terms of accuracy and training time.
5. **Comparison and Analysis**: Compare the results and analyze the impact of feature selection on model performance.

Dataset

For this case study, we will use the Arcene dataset from the UCI Machine Learning Repository. This dataset is designed for feature selection studies with a large number of features (10,000) and a small number of samples (100).

Tasks to be Performed

1. **Data Loading and Exploration**
   - Load the Arcene dataset.
   - Explore the dataset to understand its structure (number of samples, number of features, target variable distribution).
2. **Data Preprocessing**
   - Handle any missing values if present.
   - Standardize the features to have zero mean and unit variance.
3. **PCA for Feature Selection**
   - Apply PCA to the standardized dataset.
   - Select the top K principal components that explain the most variance in the data.
4. **Model Training and Evaluation**
   - Split the dataset into training and test sets.
   - Train Naïve Bayes and SVM classifiers on the full dataset.
   - Train Naïve Bayes and SVM classifiers on the dataset with selected K features.
   - Evaluate the models using accuracy and measure the training time.
5. **Comparison and Analysis**
   - Compare the accuracy and training time of the models trained on the full dataset and on the dataset with selected K features.
   - Analyze the impact of feature selection on model performance.

Questions to be Answered

1. How does PCA help in reducing the dimensionality of the dataset?
2. What are the benefits and potential drawbacks of using PCA for feature selection?

3. How does the performance (accuracy and training time) of Naïve Bayes and SVM classifiers compare when trained on the full dataset versus the dataset with selected K features?
4. What insights can be drawn from the comparison of model performance with and without feature selection?

Practical 6: Practical Implementation of K-means and DBSCAN Clustering

Aim:

To understand and implement K-means and DBSCAN clustering algorithms on a given dataset, compare their performances, and interpret the results.

Objectives

1. To learn the theoretical foundations of K-means and DBSCAN clustering algorithms.
2. To implement K-means and DBSCAN on a real-world dataset.
3. To compare the clustering results of K-means and DBSCAN.
4. To analyze the strengths and weaknesses of each clustering method.
5. To interpret and visualize the clustering outcomes.

Dataset

For this practical implementation, we will use the Iris dataset. It consists of 150 observations of iris flowers, with 4 features (sepal length, sepal width, petal length, and petal width) and 3 species (Setosa, Versicolour, and Virginica).

Dataset Link: Iris Dataset

Description

The Iris dataset is a well-known dataset used for pattern recognition. It contains three classes of iris plants, with 50 samples each. Each sample has four features: sepal length, sepal width, petal length, and petal width. The goal is to use K-means and DBSCAN to cluster the data points and analyze the results.

Tasks to be Performed During Practical Implementation

1. **Data Preprocessing**
   - Load the dataset.
   - Perform data cleaning if necessary.
   - Normalize the data to ensure all features contribute equally to the distance calculations.
2. **K-means Clustering**
   - Implement the K-means algorithm.
   - Choose an appropriate number of clusters (k) using the Elbow method or Silhouette analysis.
   - Fit the K-means model to the dataset.
   - Visualize the clustering results.
3. **DBSCAN Clustering**
   - Implement the DBSCAN algorithm.
   - Choose appropriate values for epsilon (ε) and the minimum number of samples (minPts) based on the data distribution.

- Fit the DBSCAN model to the dataset.
- Visualize the clustering results.

4. **Comparison and Analysis**
   - Compare the clustering results of K-means and DBSCAN.
   - Evaluate the performance of both algorithms using metrics such as silhouette score, Davies-Bouldin index, etc.
   - Discuss the strengths and weaknesses of each algorithm in the context of the Iris dataset.

Questions to be Answered After Completing the Practical

1. What are the main differences between K-means and DBSCAN clustering algorithms?
2. How do you determine the optimal number of clusters for K-means?
3. What criteria did you use to select the values of ε and minPts for DBSCAN?
4. How did the clustering results of K-means and DBSCAN differ for the Iris dataset?
5. Which algorithm performed better in this case study and why?
6. What are some potential real-world applications of K-means and DBSCAN?
7. How do outliers affect the performance of K-means and DBSCAN?
8. What are the advantages of using DBSCAN over K-means for clustering tasks?

Practical 7: Neural Network with Keras

Aim:

To understand and implement a Neural Network with Pima Indians Diabetes dataset using Keras.

Objectives

1. To familiarize with the Keras library for building neural networks.
2. To understand the architecture of a simple neural network.
3. To implement, train, and evaluate a neural network model.
4. To visualize the performance of the model.
5. To answer key questions about the neural network's performance and optimization.

Description

"Pima Indians Diabetes" typically refers to a well-documented case study and research focus on the high prevalence of type 2 diabetes among the Pima Native American tribe of Arizona. This population has been extensively studied due to their disproportionately high rates of diabetes, which has provided insights into genetic predispositions, lifestyle factors, and public health implications for diabetes prevention and management globally.

Dataset Link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Tasks to be Performed

1. **Loading and Preprocessing the Data**

- Load the MNIST dataset.
- Normalize the pixel values to be between 0 and 1.
- Reshape the data to fit the neural network input requirements.

2. **Building the Neural Network Model**
   - Define the architecture of the neural network (input layer, hidden layers, output layer).
   - Compile the model by specifying the optimizer, loss function, and metrics.
3. **Training the Model**
   - Split the data into training and validation sets.
   - Train the model using the training data and validate it on the validation data.
4. **Evaluating the Model**
   - Evaluate the model on the test data.
   - Calculate and print the accuracy, precision, recall, and F1 score.
5. **Visualizing the Results**
   - Plot the training and validation loss over epochs.
   - Plot the training and validation accuracy over epochs.
6. **Saving and Loading the Model**
   - Save the trained model to a file.
   - Load the saved model and use it for making predictions.

Questions to be Answered

1. **Data Understanding and Preprocessing**
   - What is the shape of the input data after preprocessing?
   - Why is it important to normalize the pixel values?
2. **Model Architecture**
   - What are the key components of the neural network model you built?
   - How does the choice of activation functions impact the performance of the model?
3. **Training Process**
   - How many epochs were needed to achieve a satisfactory level of accuracy?
   - What is the role of the validation set during training?
4. **Model Evaluation**
   - What is the final test accuracy of your model?
   - How do precision, recall, and F1 score complement the accuracy metric?
5. **Visualization and Analysis**
   - How do the training and validation loss curves help in diagnosing overfitting or underfitting?
   - What observations can you make from the training and validation accuracy curves?
6. **Model Persistence**
   - Why is it important to save the trained model?
   - How can you load a saved model and use it for making predictions?

Practical 8: Implementing Q-Learning Algorithm in Reinforcement Learning

Aim:

To understand and implement the Q-Learning algorithm for solving a reinforcement learning problem and evaluate its performance.

Objectives

1. To gain practical knowledge of Q-Learning algorithm.
2. To implement Q-Learning in a simulated environment.
3. To analyze the performance of the Q-Learning algorithm.
4. To understand the impact of different hyperparameters on the algorithm's performance.

Description

A growing e-commerce company is building a new warehouse, and the company would like all of the picking operations in the new warehouse to be performed by warehouse robots. In the context of e-commerce warehousing, "picking" is the task of gathering individual items from various locations in the warehouse in order to fulfill customer orders. After picking items from the shelves, the robots must bring the items to a specific location within the warehouse where the items can be packaged for shipping. In order to ensure maximum efficiency and productivity, the robots will need to learn the shortest path between the item packaging area and all other locations within the warehouse where the robots are allowed to travel.

Tasks to be Performed

1. **Setup Environment**:
   - Install the OpenAI Gym toolkit.
   - Load the Frozen Lake environment.
   - Understand the environment's state and action space.
2. **Implement Q-Learning Algorithm**:
   - Initialize the Q-table with zeros.
   - Define the hyperparameters: learning rate (alpha), discount factor (gamma), and exploration rate (epsilon).
   - Implement the Q-Learning update rule.
   - Implement an epsilon-greedy policy for action selection.
3. **Training the Agent**:
   - Run the Q-Learning algorithm for a fixed number of episodes.
   - Update the Q-values based on the experiences gained by the agent.
   - Monitor the agent's performance over time.
4. **Evaluation**:
   - Test the trained Q-Learning agent in the environment.
   - Measure the agent's performance using metrics such as average reward per episode.
5. **Analysis**:
   - Analyze how different hyperparameters affect the learning process.
   - Compare the performance of the Q-Learning agent with other baseline methods if available.

Questions to be Answered After Completing the Practical

1. How does the Q-Learning algorithm update the Q-values?

2. What is the role of the learning rate (alpha) and how does it affect the learning process?
3. How does the discount factor (gamma) influence the Q-Learning algorithm?
4. What is the epsilon-greedy policy and why is it used in Q-Learning?
5. How does the exploration rate (epsilon) impact the agent's learning and performance?
6. What challenges did you encounter while implementing the Q-Learning algorithm and how did you address them?
7. How does the performance of the Q-Learning agent change with different values of alpha, gamma, and epsilon?
8. Can the Q-Learning algorithm be applied to other environments? If yes, what changes would be required in the implementation?
9. Discuss the importance of choosing the right hyper parameters for the Q-Learning algorithm.
10. How does the Q-Learning algorithm compare to other reinforcement learning algorithms?
11. Analyse the reward value for the different actions.

Practical 9: Implementing a Convolutional Neural Network (CNN) for DOG vs CAT Classification

Aim:

The aim of this case study is to implement a Convolutional Neural Network (CNN) for the binary classification of images of dogs and cats. The goal is to build a model that can accurately distinguish between images of dogs and cats.

Objectives:

1. Understand the architecture and working principles of Convolutional Neural Networks.
2. Preprocess the dataset to make it suitable for training and testing the CNN.
3. Implement and train a CNN using a popular deep learning framework.
4. Evaluate the performance of the trained CNN model.
5. Optimize the model to improve its accuracy.
6. Answer key questions related to the implementation and results of the CNN.

Description:

In this case study, we will use the Kaggle Dogs vs. Cats dataset, which consists of 25,000 labeled images of dogs and cats. The dataset can be downloaded from the following link:

https://www.kaggle.com/c/dogs-vs-cats/data

Dataset:

- **Training set**: 25,000 images (12,500 dogs and 12,500 cats)
- **Test set**: To be split from the training set or use the provided test set for evaluation.

Tasks to be Performed:

1. **Dataset Preparation:**
    - Download and extract the dataset.
    - Split the dataset into training and validation sets.
    - Preprocess the images (resizing, normalization, and augmentation).
2. **Model Building:**
    - Define the architecture of the CNN.
    - Choose appropriate layers (convolutional layers, pooling layers, dense layers, etc.).
    - Implement the model using a deep learning framework (e.g., TensorFlow/Keras, PyTorch).
3. **Model Training:**
    - Compile the model with appropriate loss function and optimizer.
    - Train the model on the training set and validate it on the validation set.
    - Monitor the training process using metrics like accuracy and loss.
4. **Model Evaluation:**
    - Evaluate the model's performance on the test set.
    - Analyze the results using confusion matrix, precision, recall, and F1-score.
5. **Model Optimization:**
    - Implement techniques to improve model accuracy (e.g., hyperparameter tuning, regularization, dropout).
    - Retrain and evaluate the optimized model.
6. **Model Deployment (Optional):**
    - Save the trained model.
    - Implement a simple application to use the model for real-time image classification.

Questions to be Answered After Completing the Practical:

1. **Understanding the Dataset:**
    - How is the dataset structured, and what preprocessing steps were necessary?
    - What challenges did you encounter during data preprocessing?
2. **Model Architecture:**
    - What architecture did you choose for the CNN, and why?
    - How did you decide on the number of layers and their types?
3. **Training Process:**
    - What loss function and optimizer were used, and why?
    - How did you split the dataset for training and validation?
    - What metrics were used to monitor the training process?
4. **Model Performance:**
    - What was the accuracy of the model on the validation and test sets?
    - What do the confusion matrix and other evaluation metrics indicate about the model's performance?
5. **Optimization Techniques:**
    - What optimization techniques did you implement, and how did they affect the model's performance?
    - What were the best hyperparameters found during tuning?

6. **Deployment and Application:**
    - How can the trained model be deployed for practical use?
    - What are the potential applications of the model in real-world scenarios?
7. **Reflection:**
    - What were the main challenges you faced during the implementation of the CNN?
    - How would you improve the project if given more time or resources?

Practical's for fast learner:

The following projects encourage independent learning and exploration of Machine Learning. Student may select any one of the problem definition and complete till the semester end.

- **NLP:** Develop a document similarity detection system using Natural Language Processing that can accurately measure the similarity among multiple text documents. Design a website to perform this task which takes as an input documents and show list of the documents which are copied from, with individual similarity score and average similarity score. Initially, students can try with two documents and finally with multiple documents.

- **Reinforcement Learning:** Develop a reinforcement learning (RL) system for a retail store to optimize its inventory management, balancing stock levels to minimize costs and maximize customer satisfaction.

Scenario: Imagine a retail store selling various products with fluctuating demand. Holding excessive stock incurs storage costs and risks obsolescence, while understocking leads to lost sales and customer dissatisfaction. The goal is to design an RL agent that learns to automatically adjust inventory levels based on real-time data (e.g., sales, promotions, seasonality) to achieve the optimal balance.

- **GAN:** Develop a creative image manipulation system using Generative Adversarial Networks (GANs) to seamlessly blend one's facial features onto the iconic face of the Hulk (It can be any famous character).

Inputs: Real Face Image: A portrait image of the user's face with proper lighting and neutral expression. *Style Image*: A reference image of the Hulk's face highlighting the desired facial features and expressions. *Outputs*: Hulkified Face Image: A photorealistic image where the user's facial features and details are seamlessly grafted onto the Hulk's face, preserving the distinctive characteristics of both individuals.

- **CNN:** Challenge Design and train a Convolutional Neural Network (CNN) for a self-driving car to accurately classify objects (pedestrians, vehicles, traffic signs) in real-time from camera input, enabling safe navigation on the road.

Scenario: Imagine a self-driving car constantly receiving images from its camera sensors. The car needs to instantly identify objects in its surroundings with high accuracy to make appropriate driving decisions. This project focuses on developing a robust CNN architecture that can classify objects like pedestrians, cars, bicycles, traffic lights, and road signs under various weather and lighting conditions.