**WEEKLY ASSESMENT**

**QUESTIONS**

1. Using a deep learning framework of your choice (TensorFlow, PyTorch, etc.), implement a CNN to classify images from the CIFAR-10 dataset. Ensure your network includes convolutional layers, pooling layers, and fully connected layers. Evaluate the performance of your model and discuss any improvements you could make.

2. Construct a feedforward neural network to predict housing prices based on the provided dataset. Include input normalization, hidden layers with appropriate activation functions, and an output layer. Train the network using backpropagation and evaluate its performance using Mean Squared Error (MSE).

**ANSWERS**

**1.**

**REQUIREMENTS**

1.COMPUTER

2.VS CODE

**PROCEDURE**

1. Open vs code
2. Create a folder EXAM
3. Create Q1.py in the folder
4. Write code in Q1.py

CODE

import tensorflow as tf

from tensorflow.keras import layers, models, datasets

# Load CIFAR-10 dataset

(train\_images, train\_labels), (test\_images, test\_labels) = datasets.cifar10.load\_data()

train\_images, test\_images = train\_images / 255.0, test\_images / 255.0

# Define the CNN model

model = models.Sequential([

    layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.Flatten(),

    layers.Dense(64, activation='relu'),

    layers.Dense(10, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

# Train the model

history = model.fit(train\_images, train\_labels, epochs=10, batch\_size=64,

                    validation\_data=(test\_images, test\_labels))

# Evaluate the model

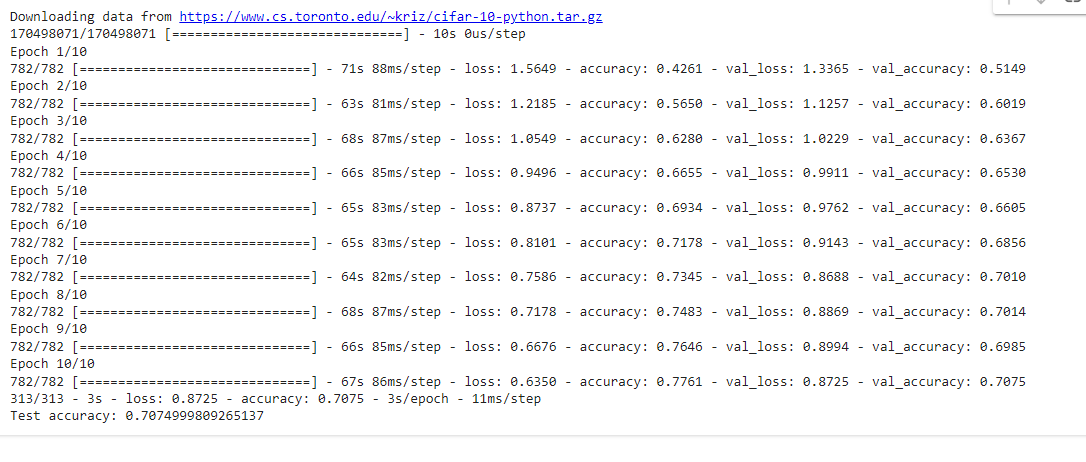
test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

print(f'Test accuracy: {test\_acc}')

.install require libraries

.run the file using py q1.py

**OUTPUT**



**2.**

**REQUIREMENTS**

1.COMPUTER

2.VS CODE

**PROCEDURE**

1. Open vs code
2. Create a folder EXAM
3. Create Q2.py in the folder
4. Write code in Q1.py

CODE

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

# Load data

try:

    data = pd.read\_csv('housing\_prices.csv')

except FileNotFoundError as e:

    print(f"FileNotFoundError: {e}. Make sure the file path is correct and accessible.")

    exit()

# Check if 'Price' column exists

if 'Price' not in data.columns:

    print("Error: 'Price' column not found in the dataset.")

    print("Columns available in the dataset:")

    print(data.columns)

    exit()

# Separate features and target

X = data.drop('Price', axis=1)

y = data['Price']

# Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Define column transformer for preprocessing

numeric\_features = ['Bedrooms', 'Bathrooms', 'SquareFootage', 'Age']

categorical\_features = ['Location']

numeric\_transformer = Pipeline(steps=[

    ('scaler', StandardScaler())

])

categorical\_transformer = Pipeline(steps=[

    ('onehot', OneHotEncoder(handle\_unknown='ignore'))

])

preprocessor = ColumnTransformer(

    transformers=[

        ('num', numeric\_transformer, numeric\_features),

        ('cat', categorical\_transformer, categorical\_features)

    ])

# Fit and transform the training data

X\_train\_processed = preprocessor.fit\_transform(X\_train)

X\_test\_processed = preprocessor.transform(X\_test)

# Determine the input shape

input\_shape = X\_train\_processed.shape[1]

# Define the neural network

model = Sequential([

    Dense(64, activation='relu', input\_shape=(input\_shape,)),

    Dense(64, activation='relu'),

    Dense(1)  # Output layer for regression, no activation function

])

# Compile the model

model.compile(optimizer=Adam(), loss='mean\_squared\_error')

# Train the model

history = model.fit(X\_train\_processed, y\_train, epochs=50, batch\_size=8, validation\_split=0.2)

# Evaluate the model

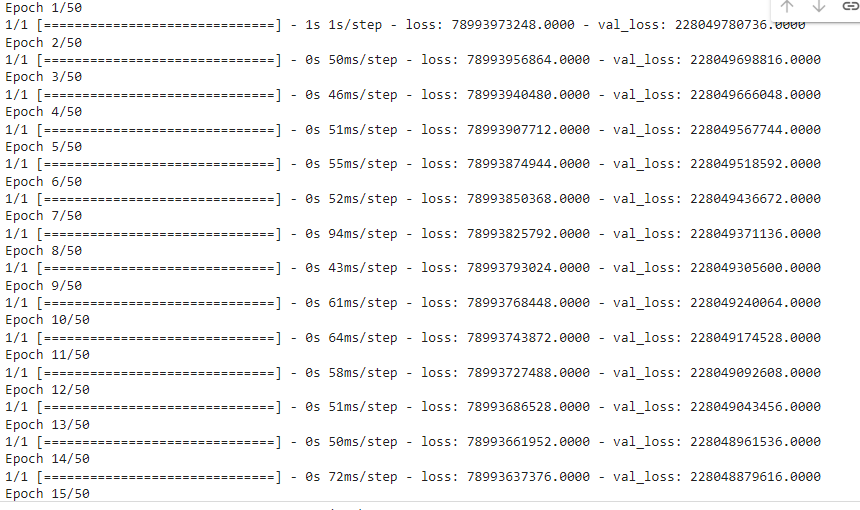
mse = model.evaluate(X\_test\_processed, y\_test)

print(f'Mean Squared Error on test set: {mse}')

.install require libraries

.run the file using py q2.py

**OUTPUT**



SHIBNU M