**DL**

Q1 **House Price Prediction using Linear Regression** project:

**Conceptual Questions with Answers**

**1. What is Linear Regression?**  
Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship, represented by a straight line.

**2. Why did you choose Linear Regression for this problem?**  
House price prediction is a regression problem where the target is a continuous value. Linear regression is simple, interpretable, and effective when there is a linear relationship between variables.

**3. What are the assumptions of Linear Regression?**

* Linearity between dependent and independent variables
* Homoscedasticity (constant variance of errors)
* No multicollinearity among features
* Normal distribution of residuals
* Independence of observations

**4. What are the dependent and independent variables in your model?**

* Dependent Variable: Price
* Independent Variables: Avg. Area Income, Avg. Area House Age, Avg. Area Number of Rooms, Avg. Area Number of Bedrooms, Area Population

**5. What is the role of the intercept and coefficients?**  
The intercept represents the baseline value of the dependent variable. Coefficients show how much the price changes with a unit change in the respective independent variable.

**6. What is R-squared?**  
R² (coefficient of determination) measures how much of the variance in the target variable is explained by the model. Values closer to 1 indicate better fit.

**7. Difference between correlation and regression?**  
Correlation measures the strength of the linear relationship between two variables, while regression predicts the value of one variable based on another.

**8. How do you evaluate model performance?**  
Using metrics like:

* Mean Absolute Error (MAE)
* Mean Squared Error (MSE)
* Root Mean Squared Error (RMSE)
* R-squared

**9. What is overfitting and underfitting?**

* Overfitting: Model fits training data too well but performs poorly on unseen data.
* Underfitting: Model is too simple to capture the pattern in the data.

**10. What is the cost function in Linear Regression?**  
Mean Squared Error (MSE): It calculates the average squared difference between predicted and actual values.

**Q2] MNIST**

**📘 Viva/External Questions & Answers**

**1. What is MNIST dataset?**

**Answer:** MNIST (Modified National Institute of Standards and Technology) is a dataset of 70,000 handwritten digits (0–9), commonly used for training image processing systems and machine learning models.

**2. Why do we normalize pixel values?**

**Answer:** Pixel values range from 0 to 255. Normalizing them between 0 and 1 helps the model converge faster and improves training stability.

**3. Why do we reshape the input to (28,28,1)?**

**Answer:** CNNs expect 3D input. Since MNIST images are grayscale, we add one channel dimension to make it (28x28x1).

**4. What is the role of Conv2D and MaxPooling2D layers?**

**Answer:**

* Conv2D extracts local features using filters.
* MaxPooling2D reduces spatial dimensions, making the model computationally efficient and reducing overfitting.

**5. What is one-hot encoding and why is it used?**

**Answer:** It converts class labels (like 3) into binary vectors (like [0 0 0 1 0 0 0 0 0 0]). This is required for the categorical\_crossentropy loss function.

**6. What does softmax do in the final layer?**

**Answer:** Softmax converts logits into probabilities that sum to 1, suitable for multiclass classification.

**7. What is a confusion matrix and what does it show?**

**Answer:** It shows the number of correct and incorrect predictions for each class, helping visualize model performance in detail.

**8. How can overfitting be reduced in CNN?**

**Answer:** Techniques include dropout, data augmentation, early stopping, and using regularization like L2.

**The reshape()** function in NumPy (or TensorFlow/Keras backend) is used to **change the shape of an array** without changing its data.

 -1: This tells Python to **automatically figure out the number of samples** (here it's 60,000).

 28: Height of image.

 28: Width of image.

 1: Number of **channels** → 1 means **grayscale**.

**. Sequential()**:

* Sequential() defines the model as a **linear stack of layers**. It means the layers will be stacked one after the other in a sequence.

**. Sequential()**:

* Sequential() defines the model as a **linear stack of layers**. It means the layers will be stacked one after the other in a sequence.

**b. Conv2D(32, kernel\_size=(3,3), activation='relu', input\_shape=(28,28,1))**:

* **Conv2D(32, kernel\_size=(3,3))**: This is the first **Convolutional Layer** that applies 32 filters (also called kernels) of size 3x3 to the input image (28x28x1).
  + **Why filters?** Filters (kernels) are used to extract features from the image (e.g., edges, textures, patterns). The number 32 represents the number of different filters used for this operation.
  + **kernel\_size=(3,3)**: A 3x3 filter slides over the image to compute convolution.
* **activation='relu'**: **ReLU (Rectified Linear Unit)** activation function introduces non-linearity to the model. It replaces all negative values with zero, which helps the model learn complex patterns.
* **input\_shape=(28,28,1)**: This defines the input image shape (28x28 pixels and 1 channel, as it's grayscale).

**c. MaxPooling2D(pool\_size=(2,2))**:

* **MaxPooling2D** reduces the spatial size (height & width) of the feature maps produced by the convolution operation.
  + **pool\_size=(2,2)**: This means it will take the maximum value from a 2x2 block of pixels. This helps in reducing the number of parameters, speeding up computation, and avoiding overfitting.

**d. Conv2D(64, kernel\_size=(3,3), activation='relu')**:

* This is the second **Convolutional Layer**, but now with 64 filters of size 3x3. It learns more complex features after the first convolution layer.
* **activation='relu'**: ReLU activation is applied again for non-linearity.

**e. MaxPooling2D(pool\_size=(2,2))**:

* Similar to the first pooling layer, this performs **max pooling** to reduce the spatial dimensions of the output from the second convolution layer.

**f. Flatten()**:

* **Flatten** converts the 2D matrix of features into a **1D vector**.
  + For example, the output from the convolutional and pooling layers is a 2D matrix, but the fully connected layers (Dense layers) require a 1D vector as input.
  + So, Flatten() takes all the 2D feature maps and turns them into one long list of values.

**g. Dense(128, activation='relu')**:

* **Dense** is a fully connected layer, meaning every neuron is connected to every neuron in the previous layer.
* **128**: This layer has 128 neurons.
* **activation='relu'**: ReLU activation function is applied to introduce non-linearity.

**h. Dense(10, activation='softmax')**:

* This is the final output layer, which has **10 neurons** (one for each class in MNIST).
* **activation='softmax'**: Softmax activation function converts the raw predictions (logits) into probabilities. Each output value is between 0 and 1, and the sum of all outputs is 1.
  + It’s used for **multiclass classification**, where the model needs to predict which of the 10 possible classes (digits 0-9) the image belongs to.