Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	Date of Submission	Submitted on:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

Convolutional Neural Network (CNN)

This tutorial demonstrates training a simple Convolutional Neural Network (CNN) to classify CIFAR images. Because this tutorial uses the Keras Sequential API, creating and training your model will take just a few lines of code.

Import TensorFlow

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

Download and prepare the CIFAR10 dataset

The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them.

```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

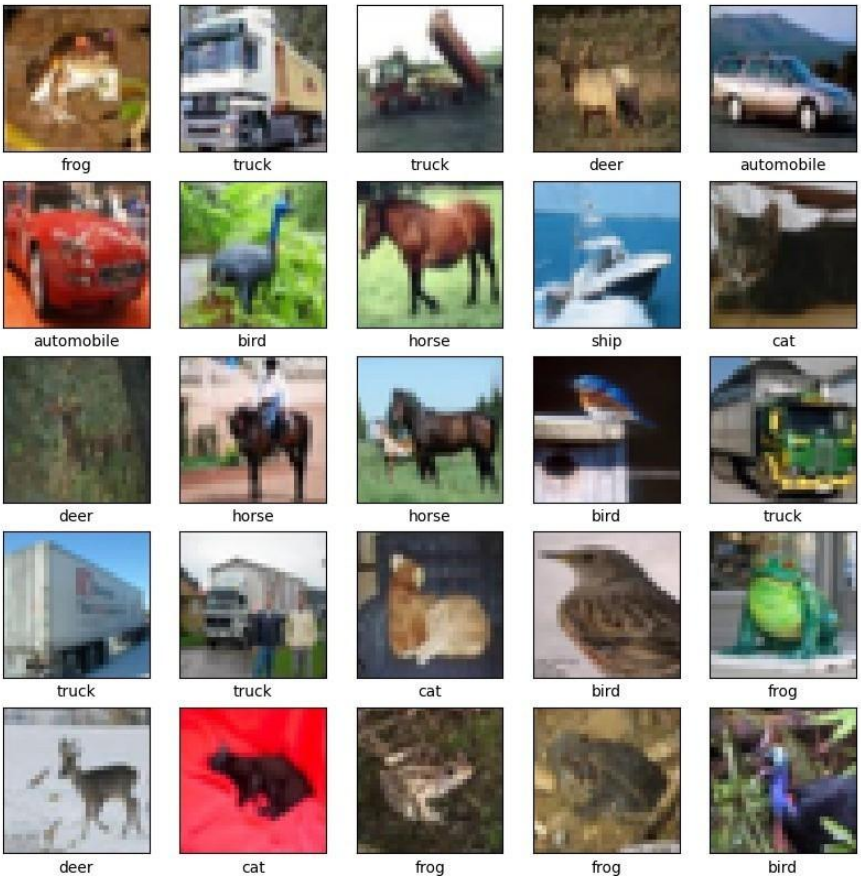
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 13s 0us/step
```

Verify the data

To verify that the dataset looks correct, let's plot the first 25 images from the training set and display the class name below each image:

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])


plt.show()
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	Date of Submission	Submitted on:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	



Create the convolutional base

The 6 lines of code below de ne the convolutional base using a common pattern: a stack of Conv2D and MaxPooling2D layers.

As input, a CNN takes tensors of shape (image_height, image_width, color_channels), ignoring the batch size. If you are new to these dimensions, color_channels refers to (R,G,B). In this example, you will con gure your CNN to process inputs of shape (32, 32, 3), which is the format of CIFAR images. You can do this by passing the argument input_shape to your rst layer.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Let's display the architecture of your model so far:

```
model.summary()
```

Above, you can see that the output of every Conv2D and MaxPooling2D layer is a 3D tensor of shape (height, width, channels). The width and height dimensions tend to shrink as you go deeper in the network. The number of output channels for each Conv2D layer is controlled by the rst argument (e.g., 32 or 64). Typically, as the width and height shrink, you can afford (computationally) to add more output channels in each Conv2D layer.

Add Dense layers on top

To complete the model, you will feed the last output tensor from the convolutional base (of shape (4, 4, 64)) into one or more Dense layers to perform classi cation. Dense layers take vectors as input (which are 1D), while the current output is a 3D tensor. First, you will atten (or unroll) the 3D output to 1D, then add one or more Dense layers on top. CIFAR has 10 output classes, so you use a nal Dense layer with 10 outputs.

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	Date of Submission	Submitted on:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

Here's the complete architecture of your model:

```
model.summary()
```

The network summary shows that (4, 4, 64) outputs were attened into vectors of shape (1024) before going through two Dense layers.

Compile and train the model

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

Evaluate the model

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

Comparative Analysis of Classification Metrics

Accuracy gives an overall picture of model performance but might be misleading in the presence of class imbalance.
Precision and recall provide insights into how well the model identifies positive instances and avoids false positives, respectively.
F1 score balances precision and recall, making it suitable for imbalanced datasets.
ROC AUC evaluates the model's ability to discriminate between classes, making it useful for assessing performance across different thresholds.

1. Dataset Selection

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. Model Training:

For simplicity, we'll train a Logistic Regression classifier using scikit-learn.

```
from sklearn.linear_model import LogisticRegression

# Initialize and train the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

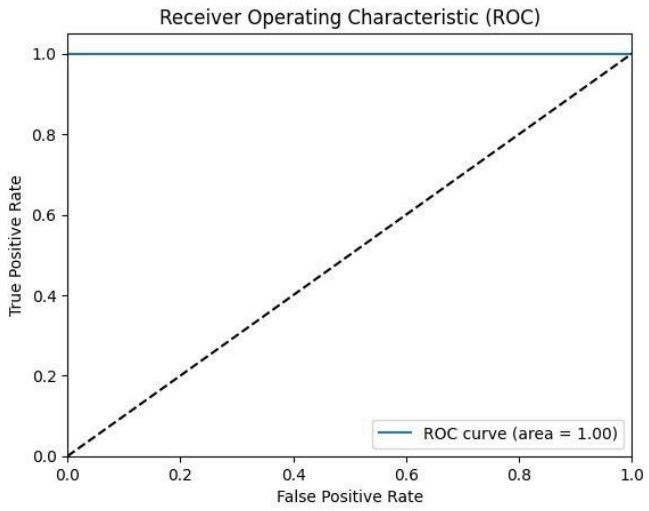
```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

3. Model Evaluation:

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Make predictions on the test set
y_pred = model.predict(X_test)

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate precision, recall, and F1 score
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Compute ROC AUC
y_probs = model.predict_proba(X_test)
roc_auc = roc_auc_score(y_test, y_probs, multi_class='ovr')

# Plot ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs[:, 1], pos_label=1)
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
plt.show()

# Print results
print("Confusion Matrix:\n", cm)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC AUC:", roc_auc)
```




```
Confusion Matrix:
[[10 0 0]
[ 0 9 0]
 [ 0 0 11]]
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

4. Summary:

- For balanced datasets like Iris, accuracy is generally reliable. However, in scenarios with class imbalance, precision, recall, and F1 score provide a better understanding of model performance.
- Precision is important when minimizing false positives is crucial, while recall is vital when capturing all positive instances is a priority.
- F1 score balances precision and recall, making it suitable for scenarios where both metrics are important.
- ROC AUC provides a comprehensive view of model performance across different thresholds and is useful when the trade-off between true positive rate and false positive rate needs to be considered.

Start coding or generate with AI.

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

Linear Regression without hyperparameter tuning

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

# Load the Diabetes dataset
X, y = load_diabetes(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the model
model = Ridge()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 3077.41593882723
```

Linear Regression with hyperparameter tuning

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

# Load the Diabetes dataset
X, y = load_diabetes(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the model
model = Ridge()

# Define the hyperparameter grid
param_grid = {'alpha': [0.01, 0.1, 1, 10, 100]}

# Perform grid search cross-validation
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Evaluate the model with the best hyperparameters
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Best Hyperparameters: {'alpha': 0.1}
Mean Squared Error: 2856.4868876706537
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

Demonstration of RNN Architecure

```
from keras import Sequential
from keras.layers import Dense, SimpleRNN


model=Sequential()
model.add(SimpleRNN(3,input_shape=(4,5)))
model.add(Dense(1,activation='sigmoid'))


model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_1 (SimpleRNN)    (None, 3)                 27

 dense_1 (Dense)             (None, 1)                 4

=================================================================
Total params: 31 (124.00 Byte)
Trainable params: 31 (124.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
print(model.get_weights()[0].shape)
model.get_weights()[0]
```

```
(5, 3)
array([[  0.59997874,  0.04102772,  -0.81552184],    [-0.01561767,
-0.15302205,  -0.35893905],   [  0.19451827,  -0.80586904,  0.7513055
],
 [  0.61303407,  0.68892676,  -0.47371438],
 [  0.21672195,  0.49013323,  0.85245126]], dtype=float32)
```

```
print(model.get_weights()[1].shape)
model.get_weights()[1]
```

```
(3, 3)
array([[ 0.08381891,  -0.06019659,     0.99466115],
        [  0.5961904 ,  -0.7967826 ,  -0.09846127],
       [-0.7984557 ,  -0.60126036,   0.03089666]], dtype=float32)
```

```
print(model.get_weights()[2].shape)
model.get_weights()[2]
```

```
(3,)
array([0., 0., 0.], dtype=float32)
```

```
print(model.get_weights()[3].shape)
model.get_weights()[3]
```

```
(3, 1)
array([][,-0.710069
       [-0.7111249  ],
       [-0.66722155]],  dtype=float32)
```

```
print(model.get_weights()[4].shape)
model.get_weights()[4]
```

```
(1,)
array([0.],  dtype=float32)
```

Integer Encoding

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
import numpy as np

docs = ['go india',
        'india india',
        'hip hip hurray',
        'jeetega bhai jeetega india jeetega',
        'bharat mata ki jai',
        'kohli kohli',
        'sachin sachin',
        'dhoni dhoni',
        'modi ji ki jai',
        'inquilab zindabad']


from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(oov_token='<nothing>')


tokenizer.fit_on_texts(docs)


tokenizer.word_index

     {'<nothing>': 1,
      'india': 2,
      'jeetega': 3,
      'hip': 4,
      'ki': 5,
      'jai': 6,
      'kohli': 7,
      'sachin': 8,
      'dhoni': 9,
      'go': 10,
      'hurray': 11,
      'bhai': 12,
      'bharat': 13,
         'mata':   14,
      'modi':      15,
      'ji':        16,
      'inquilab':  17,
      'zindabad': 18}


tokenizer.word_counts

     OrderedDict([('go', 1),
                  ('india', 4),
                  ('hip', 2),
                  ('hurray', 1),
                  ('jeetega', 3),
                  ('bhai', 1),
                  ('bharat', 1),
                  ('mata', 1),
                  ('ki', 2),
                  ('jai', 2),
                  ('kohli', 2),
                  ('sachin', 2),
                  ('dhoni', 2),
                  ('modi', 1),
                  ('ji', 1),
                  ('inquilab', 1),
                  ('zindabad', 1)])



tokenizer.document_count

     10


sequences = tokenizer.texts_to_sequences(docs)
sequences

     [[10, 2],
      [2, 2],
      [4, 4, 11],
      [3, 12, 3, 2, 3],
      [13, 14, 5, 6],
      [7, 7],
      [8, 8],
      [9, 9],
      [15, 16, 5, 6],
      [17, 18]]


from keras.utils import pad_sequences


sequences = pad_sequences(sequences,padding='post')
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
sequences
```

```
array([[21,0 ,02,0,  0], 0], 0], 3],
       [4 ,2 ,01,1 ,0,], 0], 0], 0],
     [3 ,4 ,  5,0,    0],           0]],
  [ 3, 01,2 ,0,2,   dtype=int32)
  [13, 01,4 ,5,6,
 [  7,0,7 ,  0,
 [  8,   8,  0,
 [ 9,    9,  0, 6,
  [15, 16,  0,
 [17, 18,
```

Implementation of RNN on IMDb Dataset for Sentiment Analysis

```
from keras.datasets import imdb
from keras import Sequential
from keras.layers import Dense,SimpleRNN
```

```
(X_train,y_train),(X_test,y_test)  =  imdb.load_data()
```

```
 X_train.shape
```

```
      (25000,)
```

```
X_train[0]
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
 178,
 32]
```

```
len(X_train[2])
```

```
141
```

```
X_train = pad_sequences(X_train,padding='post',maxlen=50)
X_test = pad_sequences(X_test,padding='post',maxlen=50)
```

```
X_train[0]
```

```
array([2071,  256,  141,    6, 194, 7486,  148,  226,   22,
         21,  417364, 26,  480,    5,  144,   30, 5535,   18,   51,
         36,  2248,  92,   25,  104,    4,  226,   65,   16,   38,
       1334,  1828,  16,  283,    5,   16, 4472,  113, 103,   32,
         15,   16, 5345,   19,  178,   32], dtype=int32)
```

```
model = Sequential()

model.add(SimpleRNN(32,input_shape=(50,1),return_sequences=False))
model.add(Dense(1,activation='sigmoid'))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_2 (SimpleRNN)    (None, 32)                1088

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 1121 (4.38 KB)
Trainable params: 1121 (4.38 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(X_train,y_train,epochs=5,validation_data=(X_test,y_test))
```

```
Epoch 1/5
782/782 [==============================] - 12s 13ms/step - loss: 0.6963 - accuracy: 0.5021 - val_loss: 0.6935 - val_accuracy: 0
Epoch 2/5
782/782 [==============================] - 10s 13ms/step - loss: 0.6932 - accuracy: 0.5052 - val_loss: 0.6941 - val_accuracy: 0
Epoch 3/5
782/782 [==============================] - 12s 15ms/step - loss: 0.6933 - accuracy: 0.5025 - val_loss: 0.6947 - val_accuracy: 0
Epoch 4/5
782/782 [==============================] - 10s 13ms/step - loss: 0.6927 - accuracy: 0.5078 - val_loss: 0.6944 - val_accuracy: 0
Epoch 5/5
782/782 [==============================] - 12s 16ms/step - loss: 0.6932 - accuracy: 0.4993 - val_loss: 0.6938 - val_accuracy: 0
<keras.src.callbacks.History at 0x7f3983ac0910>
```

Sentiment Predictions

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences


# Load pre-trained model
#model = tf.keras.models.load_model('sentiment_analysis_model.h5')  # Example filename

# Sample reviews
reviews = [
    "I absolutely loved this movie! The acting was superb, the storyline was captivating, and the special effects were mind-blowing
    "I was really disappointed with this film. The plot was predictable, the characters were one-dimensional, and the dialogue felt
    "What an incredible film! The performances were outstanding, especially from the lead actor who delivered a truly mesmerizing p
    "I couldn't wait for this movie to be over. The pacing was sluggish, the story was convoluted, and the ending was completely un
]


# Tokenize and pad sequences
tokenizer = Tokenizer()
tokenizer.fit_on_texts(reviews)
sequences = tokenizer.texts_to_sequences(reviews)
#max_sequence_length = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen=50) #max_sequence_length)


# Predict sentiment
predictions = model.predict(padded_sequences)

# Interpret predictions
for i, pred in enumerate(predictions):
    if pred > 0.5:
        print(f"Review {i+1}: Positive")
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

Image Processing

Three Types:

1. Black and white - pixels have the value either 0 or 1
2. Gray Scale - pixels have the value in range 0 or 255
3. Coloured Image - pixels have the value in range 0 or 255 for RED , GREEN and BLUE(RGB- primary colours)

```
import cv2
```

cv stands for computer vision. library cv2 built on numpy arrays as images are basically stored as numpy arrays

Reading an Image

```
img = cv2.imread("taj-mahal.jpg", 1)
```

second parameter 0- grayscale image 1-coloured image

```
type(img)
```

```
numpy.ndarray
```

```
img.shape
```

```
(533, 800, 3)
```

```
from google.colab.patches import cv2_imshow
```

```
cv2_imshow(img)
```

```
img.shape
```

```
(533, 800, 3)
```

```
import matplotlib.pyplot as plt
img1 = plt.imread('taj-mahal.jpg')
plt.axis('off')
plt.imshow(img1)
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
<matplotlib.image.AxesImage at 0x7f79685076a0>
```

```
img1.shape
```

```
(533, 800, 3)
```

```
img1.ndim
```

```
3
```

We can see the three RGB components of image

```
img_blue = img[:,:,0]
img_blue
```

```
img_green = img[:,:,1]
cv2_imshow(img_green)
```

```
img_red = img[:,:,2]
cv2_imshow(img_red)
```

Removing a particular color

```
img[:,:,0] = 0 #blue grid set to 0
```

```
cv2_imshow(img)
```

Blue component is removed. Similarly red and green components can also be removed

Image Resize

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
img_resize = cv2.resize(img,(150,150))
cv2_imshow(img_resize)
```

We can increase the size also.

Flipping the Image 0 - Vertical flip, 1- Horizontal flip and -1: both

```
img_flip = cv2.flip(img,0)
cv2_imshow(img_flip)
```

```
img_flip_h = cv2.flip(img,1)
cv2_imshow(img_flip_h)
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
img_flip_hv = cv2.flip(img,-1)
cv2_imshow(img_flip_hv)
```

Cropping the image, its basically array slicing operation. Image top left corner is starting point(0,0)

```
img_crop = img[0:200, 0:200]
cv2_imshow(img_crop)
```

```
img_crop1 = img[100:300, 200:400]
cv2_imshow(img_crop1)
```

Saving the image

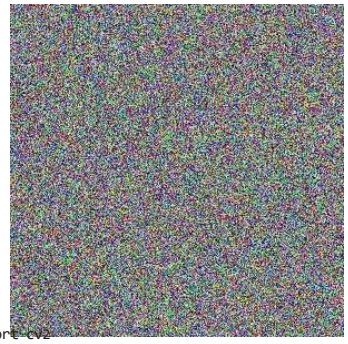
```
cv2.imwrite('taj-cropped.png', img_crop1)
```

```
True
```

creating images

```
import numpy as np
new_img = np.random.randint(255,size=(300,300,3))
cv2_imshow(new_img)
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:		Grade:
Signature of student:		Signature of Faculty:

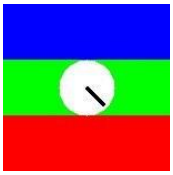


```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
im = np.ones((40,40))*150
cv2.circle(im,center=(20,20),radius=10, color = (20),thickness=-4)
cv2_imshow(im)
```

```
im
```

```
array([[255., 255., 255., ..., 255., 255., 255.],
       [255., 255., 255., ..., 255., 255., 255.], [255.,
        255., 255., ..., 255., 255., 255.], ..., [255.,
        255., 255., ..., 255., 255., 255.], [255., 255.,
        255., ..., 255., 255., 255.], [255., 255., 255.,
        ..., 255., 255., 255.]])
```

```
blue = np.ones((50,150,3)) * 255
blue[:,:,1:] = 0

green = np.ones((50,150,3)) * 255 green[:,:,0] = 0 green[:,:,2] = 0 red =
np.ones((50,150,3)) * 255 red[:,:,0:2] = 0 img_flag = np.vstack((blue,green,red))
np.vstack((blue,green,red)) cv2.circle(img_flag, center = (75,75),radius = 25, color =
(255,255,255),thickness = -1) cv2.line(img_flag, (75,75),(90,90),color = 0, thickness=2)
cv2_imshow(img_flag)
```



```
cv2.imwrite('flag.png',img_flag)
cv2.imwrite('Taj.png', img)
```

```
True
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


df = pd.read_csv('train.csv')


df.head()
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

Why do EDA

- Model building
- Analysis and reporting
- Validate assumptions
- Handling missing values
- feature engineering
- detecting outliers

```
# Remember it is an iterative process
```

Column Types

- Numerical - Age,Fare,PassengerId
- Categorical - Survived, Pclass, Gender, SibSp, Parch,Embarked
- Mixed - Name, Ticket, Cabin

Univariate Analysis

Univariate analysis focuses on analyzing each feature in the dataset independently.

- Distribution analysis: The distribution of each feature is examined to identify its shape, central tendency, and dispersion.
- Identifying potential issues: Univariate analysis helps in identifying potential problems with the data such as outliers, skewness, and missing values

The shape of a data distribution refers to its overall pattern or form as it is represented on a graph. Some common shapes of data distributions include:

- Normal Distribution: A symmetrical and bell-shaped distribution where the mean, median, and mode are equal and the majority of the data falls in the middle of the distribution with gradually decreasing frequencies towards the tails.
- Skewed Distribution: A distribution that is not symmetrical, with one tail being longer than the other. It can be either positively skewed (right-skewed) or negatively skewed (left-skewed).
  Bimodal Distribution: A distribution with two peaks or modes.
- Uniform Distribution: A distribution where all values have an equal chance of occurring.

The shape of the data distribution is important in identifying the presence of outliers, skewness, and the type of statistical tests and models that can be used for further analysis.

Dispersion is a statistical term used to describe the spread or variability of a set of data. It measures how far the values in a data set are spread out from the central tendency (mean, median, or mode) of the data.

There are several measures of dispersion, including:

- Range: The difference between the largest and smallest values in a data set.

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

- Variance: The average of the squared deviations of each value from the mean of the data set.
- Standard Deviation: The square root of the variance. It provides a measure of the spread of the data that is in the same units as the original data.
- Interquartile range (IQR): The range between the first quartile (25th percentile) and the third quartile (75th percentile) of the data.

Dispersion helps to describe the spread of the data, which can help to identify the presence of outliers and skewness in the data.

Steps of doing Univariate Analysis on Numerical columns

- Descriptive Statistics: Compute basic summary statistics for the column, such as mean, median, mode, standard deviation, range, and quartiles. These statistics give a general understanding of the distribution of the data and can help identify skewness or outliers.
- Visualizations: Create visualizations to explore the distribution of the data. Some common visualizations for numerical data include histograms, box plots, and density plots. These visualizations provide a visual representation of the distribution of the data and can help identify skewness an outliers.
- Identifying Outliers: Identify and examine any outliers in the data. Outliers can be identified using visualizations. It is important to determine whether the outliers are due to measurement errors, data entry errors, or legitimate differences in the data, and to decide whether to include or exclude them from the analysis.
- Skewness: Check for skewness in the data and consider transforming the data or using robust statistical methods that are less sensitive to skewness, if necessary.
- Conclusion: Summarize the findings of the EDA and make decisions about how to proceed with further analysis.

Age

conclusions

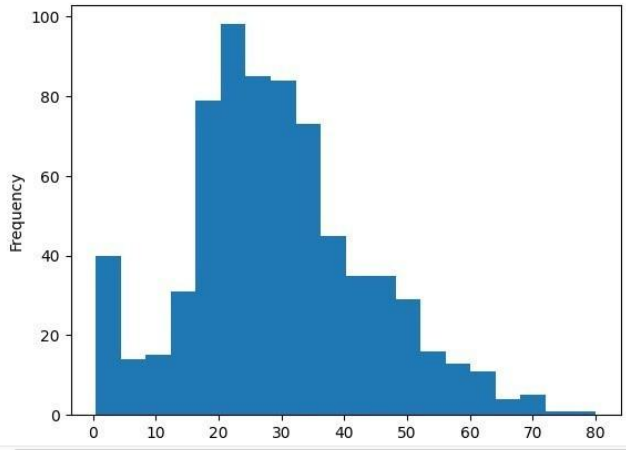
- Age is normally(almost) distributed
- 20% of the values are missing
- There are some outliers

```
df['Age'].describe()
```

```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

```
df['Age'].plot(kind='hist',bins=20)
```

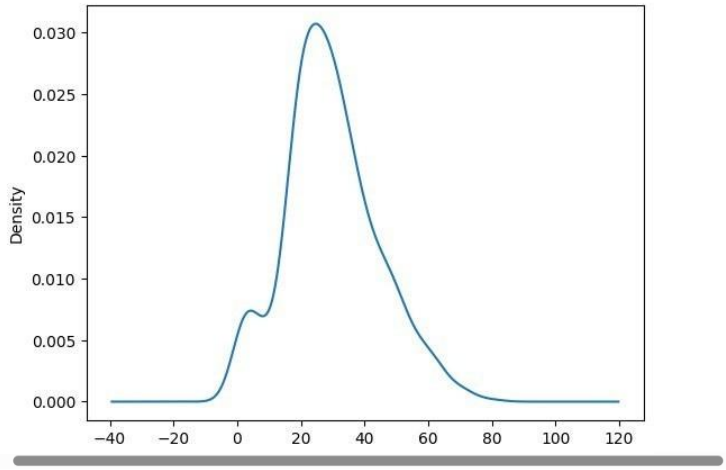
```
<Axes: ylabel='Frequency'>
```




```
df['Age'].plot(kind='kde')
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
<Axes: ylabel='Density'>
```

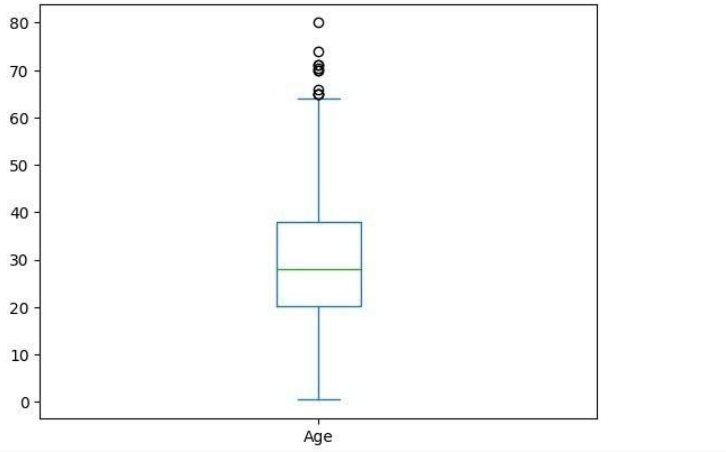



```
df['Age'].skew()
```

```
0.38910778230082704
```

```
df['Age'].plot(kind='box')
```

```
<Axes: >
```




```
df[df['Age'] > 65]
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q
493	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S
672	673	0	2	Mitchell, Mr. Henry Michael	male	70.0	0	0	C.A. 24580	10.5000	NaN	S

```
df['Age'].isnull().sum()/len(df['Age'])
```

```
0.19865319865319866
```

Fare

conclusions

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

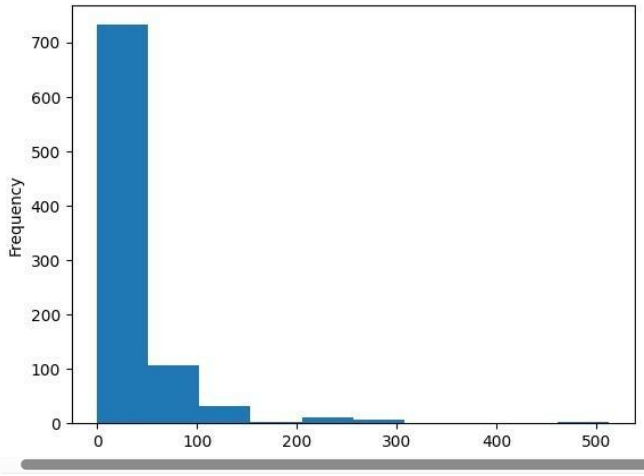
- The data is highly(positively) skewed
- Fare col actually contains the group fare and not the individual fare(This migth be and issue)
- We need to create a new col called individual fare

```
df['Fare'].describe()
```

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

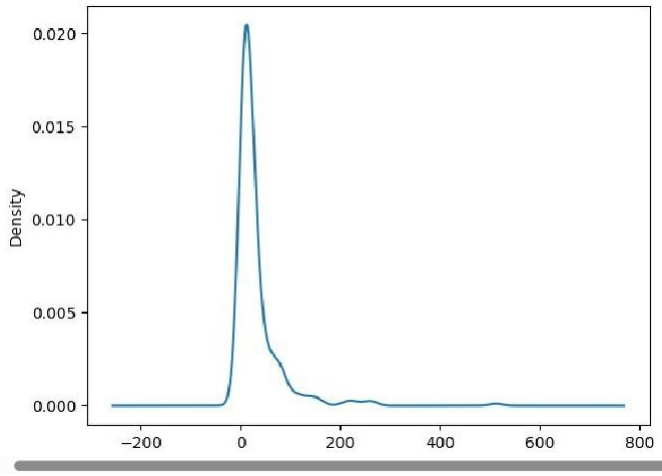
```
df['Fare'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```




```
df['Fare'].plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```



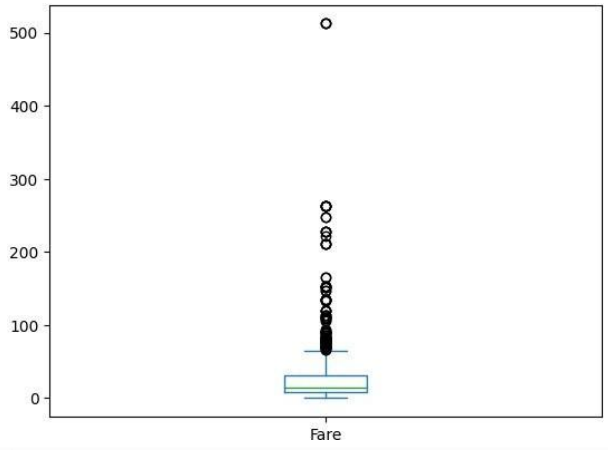

```
df['Fare'].skew()
```

```
4.787316519674893
```

```
df['Fare'].plot(kind='box')
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
<Axes: >
```




```
df[df['Fare'] > 250]
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	C
311	312	1	1	Ryerson, Miss. Emily Borie	female	18.0	2	2	PC 17608	262.3750	B57 B59 B63 B66	C
341	342	1	1	Fortune, Miss. Alice Elizabeth	female	24.0	3	2	19950	263.0000	C23 C25 C27	S
438	439	0	1	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	C23 C25 C27	S
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	177P5C5	512.3292	B51 B53 B55	C

```
df['Fare'].isnull().sum()
```

```
0
```

Steps of doing Univariate Analysis on Categorical columns

Descriptive Statistics: Compute the frequency distribution of the categories in the column. This will give a general understanding of the distribution of the categories and their relative frequencies.

Visualizations: Create visualizations to explore the distribution of the categories. Some common visualizations for categorical data include count plots and pie charts. These visualizations provide a visual representation of the distribution of the categories and can help identify any patterns or anomalies in the data.

Missing Values: Check for missing values in the data and decide how to handle them. Missing values can be imputed or excluded from the analysis, depending on the research question and the data set.

Conclusion: Summarize the findings of the EDA and make decisions about how to proceed with further analysis.

Survived

conclusions

- Parch and SibSp cols can be merged to form a new col call family_size
- Create a new col called is_alone

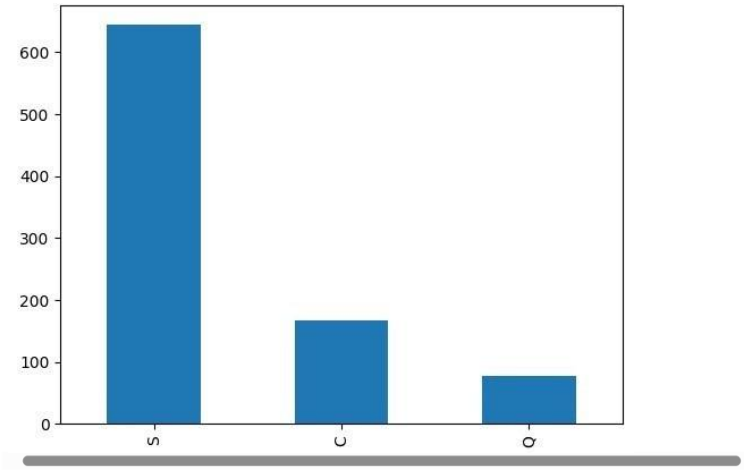
```
df['Embarked'].value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:		Grade:
Signature of student:	Signature of Faculty:	

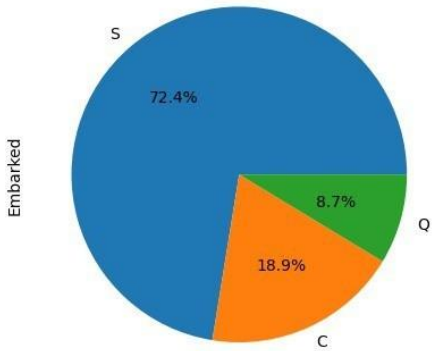

```
df['Embarked'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
df['Embarked'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

```
<Axes: ylabel='Embarked'>
```




```
df['Gender'].isnull().sum()
```

```
0
```

Steps of doing Bivariate Analysis

- Select 2 cols
- Understand type of relationship
  1. Numerical - Numerical
     a. You can plot graphs like scatterplot(regression plots), 2D histplot, 2D KDEplots
     b. Check correlation coefficent to check linear relationship
  2. Numerical - Categorical - create visualizations that compare the distribution of the numerical data across different categories of the categorical data.
     a. You can plot graphs like barplot, boxplot, kdeplot violinplot even scatterplots
  3. Categorical - Categorical
     a. You can create cross-tabulations or contingency tables that show the distribution of values in one categorical column, grouped by the values in the other categorical column.
     b. You can plots like heatmap, stacked barplots, treemaps
- Write your conclusions

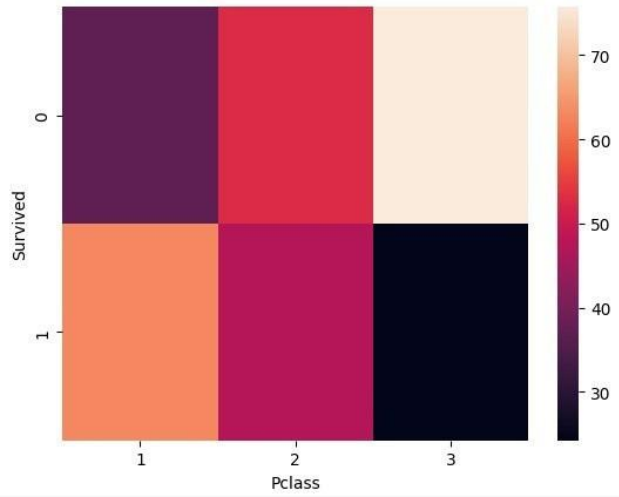
```
df
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

```
sns.heatmap(pd.crosstab(df['Survived'],df['Pclass'],normalize='columns')*100)
```

```
<Axes: xlabel='Pclass', ylabel='Survived'>
```




```
pd.crosstab(df['Survived'],df['Gender'],normalize='columns')*100
```

Gender	female	male
Survived		
0	25.796178	81.109185
1	74.203822	18.890815

```
pd.crosstab(df['Survived'],df['Embarked'],normalize='columns')*100
```

Embarked	C	Q	S
Survived			
0	44.642857	61.038961	66.304348
1	55.357143	38.961039	33.695652

```
pd.crosstab(df['Gender'],df['Embarked'],normalize='columns')*100
```

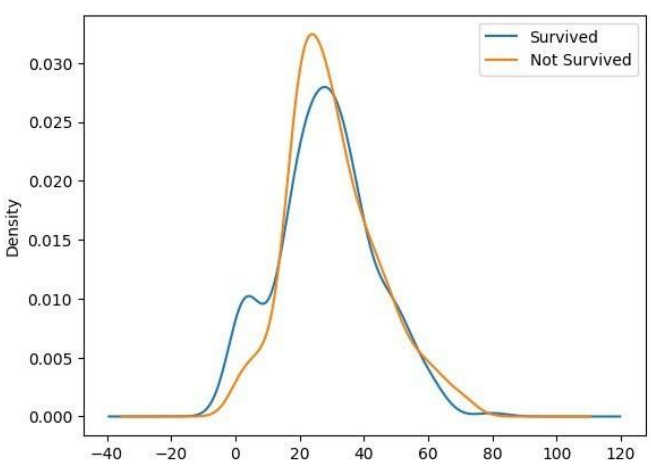
Embarked	C	Q	S
Gender			
female	43.452381	46.753247	31.521739
male	56.547619	53.246753	68.478261

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
pd.crosstab(df['Pclass'],df['Embarked'],normalize='columns')*100
```

Embarked	C	Q	S
Pclass			
1	50.595238	2.597403	19.720497
2	10.119048	3.896104	25.465839
3	39.285714	93.506494	54.813665

```
# survived and age

df[df['Survived'] == 1]['Age'].plot(kind='kde',label='Survived')
df[df['Survived'] == 0]['Age'].plot(kind='kde',label='Not Survived')

plt.legend()
plt.show()
```




```
df[df['Pclass'] == 1]['Age'].mean()
```

```
38.233440860215055
```

```
# Feature Engineering on Fare col

df['SibSp'].value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
df[df['Ticket'] == 'CA. 2343']
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8		2CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8		2CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr. Frederick	male	NaN	8		2CA. 2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8		2CA. 2343	69.55	NaN	S
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8		2CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8		2CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8		2CA. 2343	69.55	NaN	S

```
df[df['Name'].str.contains('Sage')]
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8	2	CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8	2	CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr. Frederick	male	NaN	8	2	CA. 2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.55	NaN	S
641	642	1	1	Sagesser, Mlle. Emma	female	24.0	0	0	PC 17477	69.30	B35	C
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8	2	CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.55	NaN	S

```
df1 = pd.read_csv('/content/test.csv')
```

```
df = pd.concat([df,df1])
```

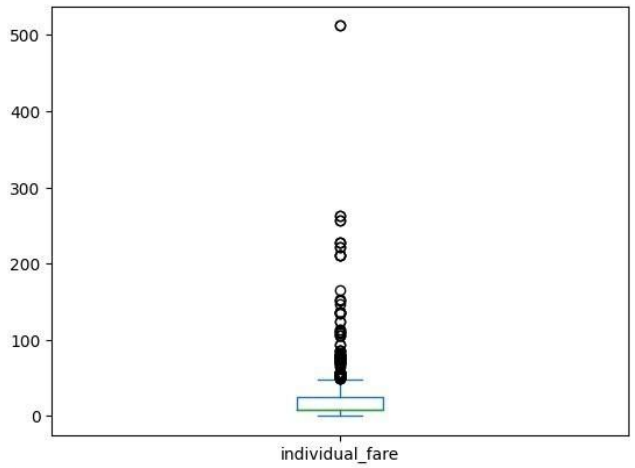
```
df[df['Ticket'] == 'CA 2144']
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
59	60	0.0	3	Goodwin, Master. William Frederick	male	11.0	5	2	CA 2144	46.9	NaN	S
71	72	0.0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.9	NaN	S
386	387	0.0	3	Goodwin, Master. Sidney Leonard	male	1.0	5	2	CA 2144	46.9	NaN	S
480	481	0.0	3	Goodwin, Master. Harold Victor	male	9.0	5	2	CA 2144	46.9	NaN	S
678	679	0.0	3	Goodwin, Mrs. Frederick (Augusta Tyler)	female	43.0	1	6	CA 2144	46.9	NaN	S
683	684	0.0	3	Goodwin, Mr. Charles Edward	male	14.0	5	2	CA 2144	46.9	NaN	S
139	1031	NaN	3	Goodwin, Mr. Charles Frederick	male	40.0	1	6	CA 2144	46.9	NaN	S
140	1032	NaN	3	Goodwin, Miss. Jessie Allis	female	10.0	5	2	CA 2144	46.9	NaN	S

```
df['individual_fare'] = df['Fare']/(df['SibSp'] + df['Parch'] + 1)
```

```
df['individual_fare'].plot(kind='box')
```

```
<Axes: >
```




```
df[['individual_fare','Fare']].describe()
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

	individual_fare	Fare
count	1308.000000	1308.000000
mean	20.518215	33.295479
std	35.774337	51.758668
min	0.000000	0.000000
25%	7.452767	7.895800
50%	8.512483	14.454200
75%	24.237500	31.275000
max	512.329200	512.329200

```
df['Fare']
```

```
0        7.2500
1       71.2833
2        7.9250
3       53.1000
4        8.0500
         ...
413      8.0500
414    108.9000
415      7.2500
416      8.0500
417     22.3583
Name: Fare, Length: 1309, dtype: float64
```

```
df
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	individual_fare
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	3.625000
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	35.641650
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	7.925000
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	26.550000
4	5	0.0	3	Allen, Mr. William	male	35.0	0	0	373450	8.0500	NaN	S	8.050000

```
df['family_size'] = df['SibSp'] + df['Parch'] + 1
```

```
# family_type
#  1  ->  alone
# 2-4 -> small
# >5 -> large
```

```
def transform_family_size(num):

  if num == 1:
    return 'alone'
  elif num>1 and num <5:
    return "small"
  else:
    return "large"
```

```
df['family_type'] = df['family_size'].apply(transform_family_size)
```

```
df
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	individual_fare	fam
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	3.625000	
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	35.641650	
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	7.925000	
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	26.550000	
				Allen, Mr.										

```
pd.crosstab(df['Survived'],df['family_type'],normalize='columns')*100
```

family_type	alone	large	small
Survived			
0.0	69.646182	83.870968	42.123288
1.0	30.353818	16.129032	57.876712

```
df['surname'] = df['Name'].str.split(',').str.get(0)
```

```
df
```

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	individual_fare	fam
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	3.625000	
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	35.641650	
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	7.925000	
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	26.550000	
				Allen, Mr. William Henry										
4	5	0.0	3	...	male	35.0	0	0	373450	8.0500	NaN	S	8.050000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

```
df['title'] = df['Name'].str.split(',').str.get(1).str.strip().str.split(' ').str.get(0)
```

```
temp_df = df[df['title'].isin(['Mr.','Miss.','Mrs.','Master.','ootherr'])]
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
pd.crosstab(temp_df['Survived'],temp_df['title'],normalize='columns')*100
```

title	Master.	Miss.	Mr.	Mrs.
Survived				
0.0	42.5	30.21978	84.332689	20.8
1.0	57.5	69.78022	15.667311	79.2

```
df['title']     =     df['title'].str.replace('Rev.','other')
df['title']     =     df['title'].str.replace('Dr.','other')
df['title']     =     df['title'].str.replace('Col.','other')
df['title']     =     df['title'].str.replace('Major.','other')
df['title']     =     df['title'].str.replace('Capt.','other')
df['title']     =     df['title'].str.replace('the','other')
df['title'] = df['title'].str.replace('Jonkheer.','other') #
,'Dr.','Col.','Major.','Don.','Capt.','the','Jonkheer.']
```

```
<ipython-input-53-ffb23deeff2c>:1: FutureWarning: The default value of regex will change from True to False in a future version.
  df['title'] = df['title'].str.replace('Rev.','other')
<ipython-input-53-ffb23deeff2c>:2: FutureWarning: The default value of regex will change from True to False in a future version.
  df['title'] = df['title'].str.replace('Dr.','other')
<ipython-input-53-ffb23deeff2c>:3: FutureWarning: The default value of regex will change from True to False in a future version.
  df['title'] = df['title'].str.replace('Col.','other')
<ipython-input-53-ffb23deeff2c>:4: FutureWarning: The default value of regex will change from True to False in a future version.
  df['title'] = df['title'].str.replace('Major.','other')
<ipython-input-53-ffb23deeff2c>:5: FutureWarning: The default value of regex will change from True to False in a future version.
  df['title'] = df['title'].str.replace('Capt.','other')
<ipython-input-53-ffb23deeff2c>:7: FutureWarning: The default value of regex will change from True to False in a future version.
  df['title'] = df['title'].str.replace('Jonkheer.','other')
```

```
df['Cabin'].isnull().sum()/len(df['Cabin'])
```

```
0.774637127578304
```

```
df['Cabin'].fillna('M',inplace=True)
```

```
df['Cabin'].value_counts()
```

```
M                   1014
C23 C25 C27            6
B57 B59 B63 B66        5
G6                     5
F33                    4
                    ...
A14                    1
E63                    1
E12                    1
E38                    1
C105                   1
Name: Cabin, Length: 187, dtype: int64
```

```
df['deck'] = df['Cabin'].str[0]
```

```
df['deck'].value_counts()
```

```
M    1014
C      94
B      65
D      46
E      41
A      22
F      21
G       5
T       1
Name: deck, dtype: int64
```

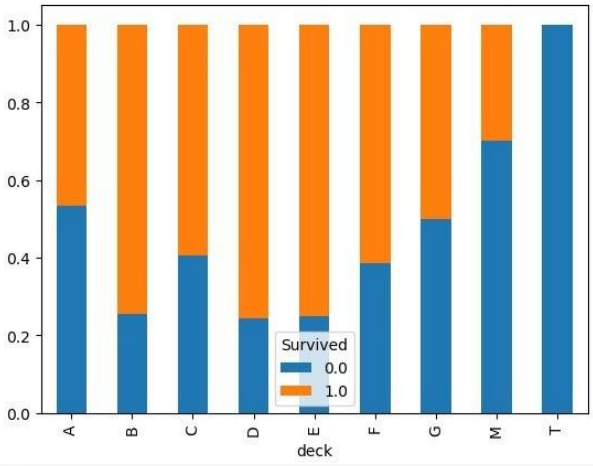
```
pd.crosstab(df['deck'],df['Pclass'])
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

Pclass	1	2	3
deck			
A	22	0	0
B	65	0	0
C	94	0	0
D	40	6	0
E	34	4	3
F	0	13	8
G	0	0	5
M	67	254	693
T	1	0	0

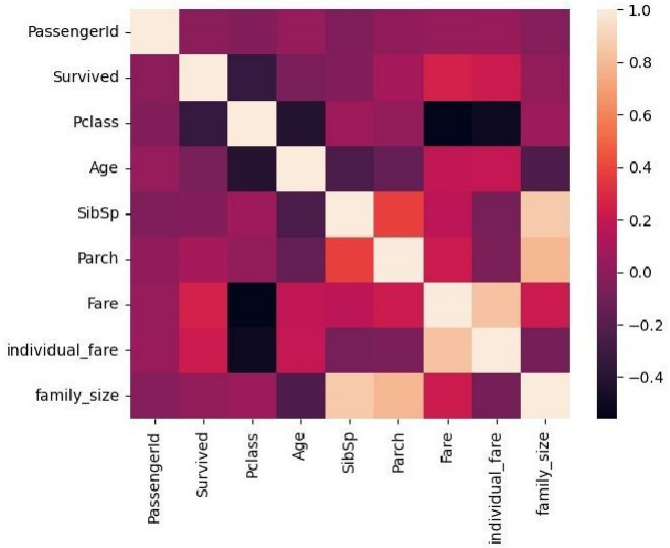
```
pd.crosstab(df['deck'],df['Survived'],normalize='index').plot(kind='bar',stacked=True)
```

```
<Axes: xlabel='deck'>
```




```
sns.heatmap(df.corr())
```

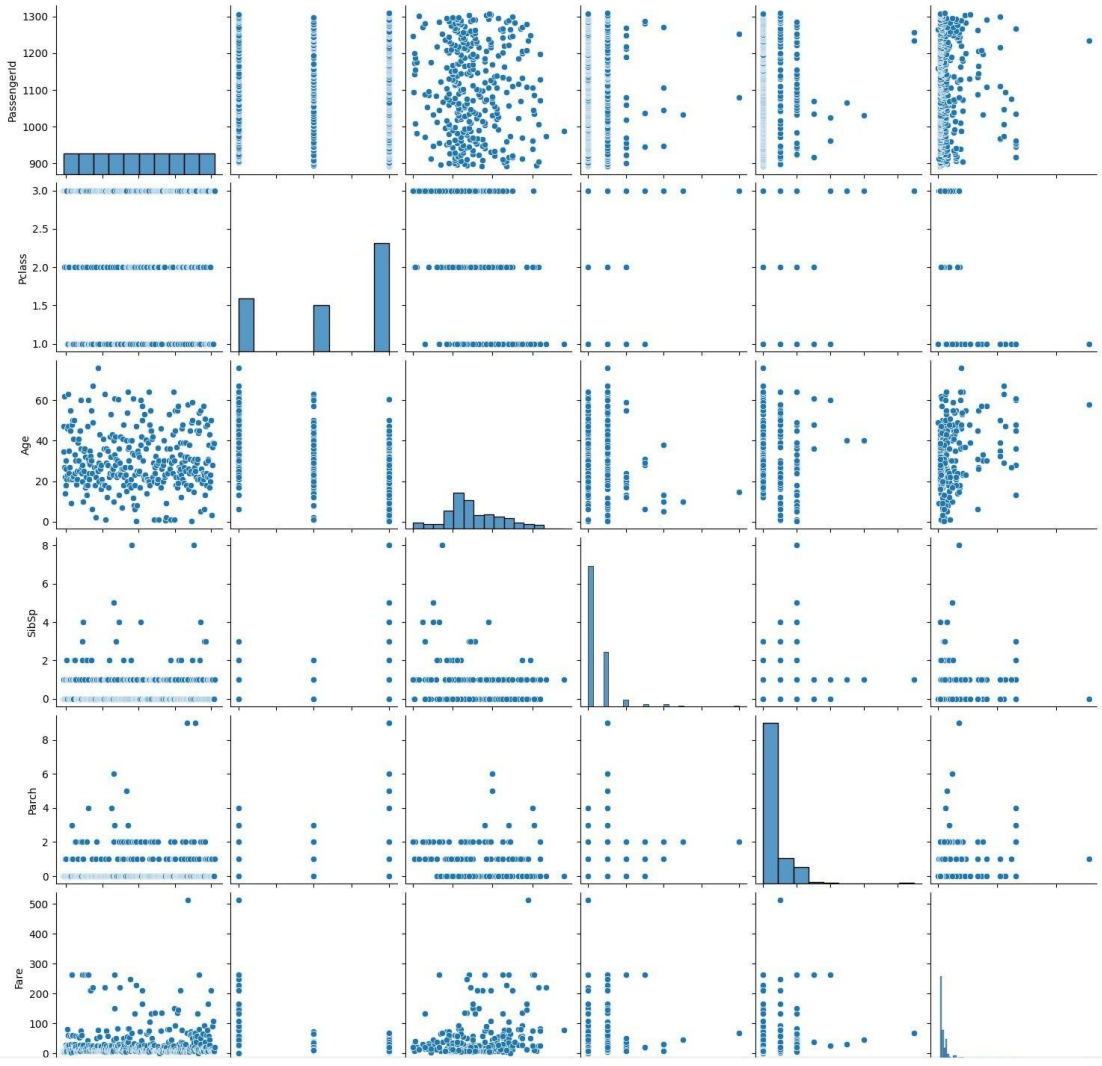
```
<ipython-input-61-aa4f4450a243>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
 sns.heatmap(df.corr())
<Axes: >
```



Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x7aa059c9f970>
```




```
df1
```

	PassengerId	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

This exercise is to implement Lenet 5 architecture using Keras Library

1. Import the required Libraries.

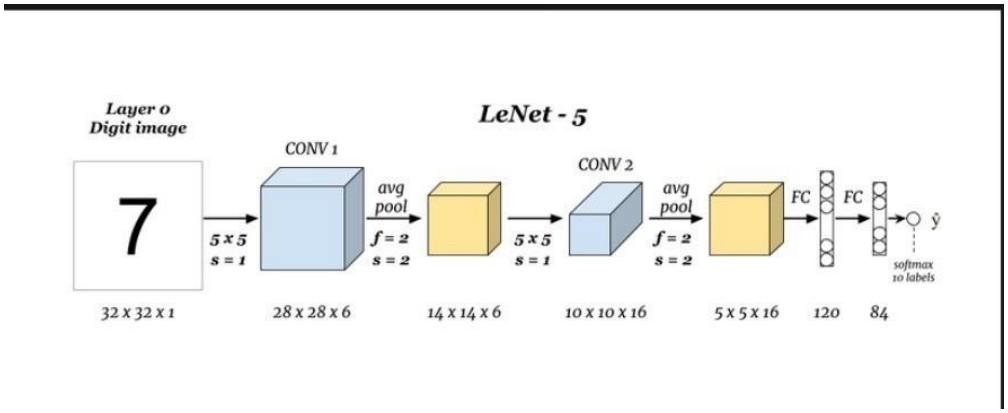
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

2. This Step involves Load the MNIST Dataset Min-Max Normalization Setting up validation dataset

```
(train_x, train_y), (test_x, test_y) = keras.datasets.mnist.load_data()
train_x = train_x / 255.0 test_x = test_x / 255.0 print(train_x.shape)
train_x = tf.expand_dims(train_x, 3) test_x = tf.expand_dims(test_x, 3)
print(train_x.shape) val_x = train_x[:5000] val_y = train_y[:5000]
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
(60000, 28, 28)
(60000, 28, 28, 1)
```

Lenet 5 Architecture




```
lenet_5_model = keras.models.Sequential([
    keras.layers.Conv2D(6, kernel_size=5, strides=1, activation='tanh', input_shape=train_x[0].shape, padding='same'), #C1
    keras.layers.AveragePooling2D(), #S2
    keras.layers.Conv2D(16, kernel_size=5, strides=1, activation='tanh', padding='valid'), #C3
    keras.layers.AveragePooling2D(), #S4
    keras.layers.Conv2D(120, kernel_size=5, strides=1, activation='tanh', padding='valid'), #C5
    keras.layers.Flatten(), #Flatten
    keras.layers.Dense(84, activation='tanh'), #F6
    keras.layers.Dense(10, activation='softmax') #Output layer
])
```

```
lenet_5_model.compile(optimizer='adam', loss=keras.losses.sparse_categorical_crossentropy, metrics=['accuracy'])
```

```
lenet_5_model.fit(train_x, train_y, epochs=5, validation_data=(val_x, val_y))
```

```
Epoch 1/5
1875/1875 [==============================] - 52s 26ms/step - loss: 0.2282 - accuracy: 0.9312 - val_loss: 0.0774 - val_accuracy: 0.97
Epoch 2/5
1875/1875 [==============================] - 50s 26ms/step - loss: 0.0837 - accuracy: 0.9740 - val_loss: 0.0639 - val_accuracy: 0.98
Epoch 3/5
1875/1875 [==============================] - 49s 26ms/step - loss: 0.0564 - accuracy: 0.9827 - val_loss: 0.0422 - val_accuracy: 0.98
Epoch 4/5
1875/1875 [==============================] - 49s 26ms/step - loss: 0.0451 - accuracy: 0.9856 - val_loss: 0.0313 - val_accuracy: 0.99
Epoch 5/5
1875/1875 [==============================] - 47s 25ms/step - loss: 0.0356 - accuracy: 0.9882 - val_loss: 0.0232 - val_accuracy: 0.99
<keras.callbacks.History at 0x7f6d31d1d0a0>
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	


```
lenet_5_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         156

 average_pooling2d (AverageP (None, 14, 14, 6)         0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_1 (Averag (None, 5, 5, 16)          0
 ePooling2D)

 conv2d_2 (Conv2D)           (None, 1, 1, 120)         48120

 flatten (Flatten)           (None, 120)               0

 dense (Dense)               (None, 84)                10164

 dense_1 (Dense)             (None, 10)                850

=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

Name of Student: Ishika Bansiwal		Class: BTech CS-2
Enrollment No: 0827CS221109		Batch: 01
Date of Experiment	DateofSubmission	Submittedon:
Remarks by faculty:	Grade:	
Signature of student:	Signature of Faculty:	

A Car Company is planning to build a Machine Learning model for predicting the buying behaviour of customer.

```
# importing libraries
import numpy as nm
import pandas as pd

#importing datasets
from google.colab import files
uploaded = files.upload()


df = pd.read_csv('User_Data.csv') #red csv file in Dataframe


df.head() #display first 5 rows of dataset
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
X = df[['Age','EstimatedSalary']] #extract independent variables
y = df['Purchased']#extract dependent variable


print(X)


print(y)


# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(X, y, test_size= 0.25, random_state=0)




#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)


print(x_train)


#Fitting Logistic Regression to the training set from
sklearn.linear_model     import     LogisticRegression
classifier=                       LogisticRegression()
classifier.fit(x_train, y_train)

     LogisticRegression()


#Predicting the test set result
y_pred= classifier.predict(x_test)


#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm= confusion_matrix(y_test,y_pred)
ac = accuracy_score(y_test,y_pred)


print(cm)
print(ac)

     [[65 3]
      [ 8 24]]
     0.89
```