HAPPYMONK TASK

Barath Vignesh.S
barathvignesh01@gmail.com

TASK 1

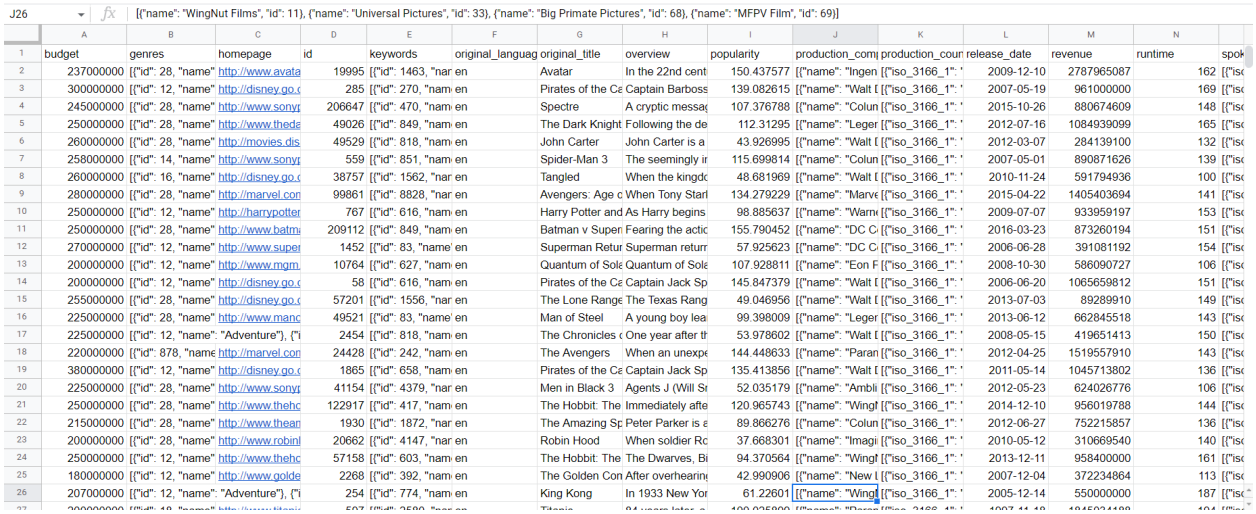
QUESTION 1

1. Consider a large dataset (say, a time series) A. Also, consider a smaller dataset B. How do you ensure whether sets A and B identify the same variable? Illustrate it with a Python script.

DATASET A - LARGE DATASET

The aim is to recommend movies on amazon prime based on voting and ratings given to each movie. The task is completed by taking 2 large dataset and combining them accordingly .

It consists of field like movie_id , title , cast ,crew ,budget ,keywords ,original_language, runtime, original_title , overview, popularity, production_companies, production_countries ,release_date, revenue, spoken_languages, status, tagline, vote_average, vote_count.



J26 [{"name": "WingNut Films", "id": 11}, {"name": "Universal Pictures", "id": 33}, {"name": "Big Primate Pictures", "id": 68}, {"name": "MFPV Film", "id": 69}]

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	budget	genres	homepage	id	keywords	original_languag	original_title	overview	popularity	production_com	production_coun	release_date	revenue	runtime	spok
2	237000000	[{"id": 28, "name"	http://www.avata	19995	[{"id": 1463, "nar	en	Avatar	In the 22nd cent	150.437577	[{"name": "Ingen	[{"iso_3166_1": '	2009-12-10	2787965087	162	[{"isc
3	300000000	[{"id": 12, "name"	http://disney.go.c	285	[{"id": 270, "nam	en	Pirates of the Ca	Captain Barboss	139.082615	[{"name": "Walt I	[{"iso_3166_1": '	2007-05-19	961000000	169	[{"isc
4	245000000	[{"id": 28, "name"	http://www.sonyp	206647	[{"id": 470, "nam	en	Spectre	A cryptic messag	107.376788	[{"name": "Colun	[{"iso_3166_1": '	2015-10-26	880674609	148	[{"isc
5	250000000	[{"id": 28, "name"	http://www.theda	49026	[{"id": 849, "nam	en	The Dark Knight	Following the de	112.31295	[{"name": "Leger	[{"iso_3166_1": '	2012-07-16	1084939099	165	[{"isc
6	260000000	[{"id": 28, "name"	http://movies.dis	49529	[{"id": 818, "nam	en	John Carter	John Carter is a	43.926995	[{"name": "Walt I	[{"iso_3166_1": '	2012-03-07	284139100	132	[{"isc
7	258000000	[{"id": 14, "name"	http://www.sonyp	559	[{"id": 851, "nam	en	Spider-Man 3	The seemingly ir	115.699814	[{"name": "Colun	[{"iso_3166_1": '	2007-05-01	890871626	139	[{"isc
8	260000000	[{"id": 16, "name"	http://disney.go.c	38757	[{"id": 1562, "nar	en	Tangled	When the kingdo	48.681969	[{"name": "Walt I	[{"iso_3166_1": '	2010-11-24	591794936	100	[{"isc
9	280000000	[{"id": 28, "name"	http://marvel.com	99861	[{"id": 8828, "nar	en	Avengers: Age o	When Tony Starl	134.279229	[{"name": "Marve	[{"iso_3166_1": '	2015-04-22	1405403694	141	[{"isc
10	250000000	[{"id": 12, "name"	http://harrypotter	767	[{"id": 616, "nam	en	Harry Potter and	As Harry begins	98.885637	[{"name": "Warn	[{"iso_3166_1": '	2009-07-07	933959197	153	[{"isc
11	250000000	[{"id": 28, "name"	http://www.batma	209112	[{"id": 849, "nam	en	Batman v Super	Fearing the actic	155.790452	[{"name": "DC C	[{"iso_3166_1": '	2016-03-23	873260194	151	[{"isc
12	270000000	[{"id": 12, "name"	http://www.super	1452	[{"id": 83, "name'	en	Superman Retur	Superman returr	57.925623	[{"name": "DC C	[{"iso_3166_1": '	2006-06-28	391081192	154	[{"isc
13	200000000	[{"id": 12, "name"	http://www.mgm.	10764	[{"id": 627, "nam	en	Quantum of Sola	Quantum of Sola	107.928811	[{"name": "Eon F	[{"iso_3166_1": '	2008-10-30	586090727	106	[{"isc
14	200000000	[{"id": 12, "name"	http://disney.go.c	58	[{"id": 616, "nam	en	Pirates of the Ca	Captain Jack Sp	145.847379	[{"name": "Walt I	[{"iso_3166_1": '	2006-06-20	1065659812	151	[{"isc
15	255000000	[{"id": 28, "name"	http://disney.go.c	57201	[{"id": 1556, "nar	en	The Lone Range	The Texas Rang	49.046956	[{"name": "Walt I	[{"iso_3166_1": '	2013-07-03	89289910	149	[{"isc
16	225000000	[{"id": 28, "name"	http://www.manc	49521	[{"id": 83, "name'	en	Man of Steel	A young boy lea	99.398009	[{"name": "Leger	[{"iso_3166_1": '	2013-06-12	662845518	143	[{"isc
17	225000000	[{"id": 12, "name": "Adventure"}, {"i		2454	[{"id": 818, "nam	en	The Chronicles o	One year after th	53.978602	[{"name": "Walt I	[{"iso_3166_1": '	2008-05-15	419651413	150	[{"isc
18	220000000	[{"id": 878, "name	http://marvel.com	24428	[{"id": 242, "nam	en	The Avengers	When an unexpe	144.448633	[{"name": "Paran	[{"iso_3166_1": '	2012-04-25	1519557910	143	[{"isc
19	380000000	[{"id": 12, "name"	http://disney.go.c	1865	[{"id": 658, "nam	en	Pirates of the Ca	Captain Jack Sp	135.413856	[{"name": "Walt I	[{"iso_3166_1": '	2011-05-14	1045713802	136	[{"isc
20	225000000	[{"id": 28, "name"	http://www.sonyp	41154	[{"id": 4379, "nar	en	Men in Black 3	Agents J (Will Sr	52.035179	[{"name": "Ambli	[{"iso_3166_1": '	2012-05-23	624026776	106	[{"isc
21	250000000	[{"id": 28, "name"	http://www.thehc	122917	[{"id": 417, "nam	en	The Hobbit: The	Immediately afte	120.965743	[{"name": "Wingf	[{"iso_3166_1": '	2014-12-10	956019788	144	[{"isc
22	215000000	[{"id": 28, "name"	http://www.thean	1930	[{"id": 1872, "nar	en	The Amazing Sp	Peter Parker is a	89.866276	[{"name": "Colun	[{"iso_3166_1": '	2012-06-27	752215857	136	[{"isc
23	200000000	[{"id": 28, "name"	http://www.robinl	20662	[{"id": 4147, "nar	en	Robin Hood	When soldier Rc	37.668301	[{"name": "Imagi	[{"iso_3166_1": '	2010-05-12	310669540	140	[{"isc
24	250000000	[{"id": 12, "name"	http://www.thehc	57158	[{"id": 603, "nam	en	The Hobbit: The	The Dwarves, Bi	94.370564	[{"name": "Wingf	[{"iso_3166_1": '	2013-12-11	958400000	161	[{"isc
25	180000000	[{"id": 12, "name"	http://www.golde	2268	[{"id": 392, "nam	en	The Golden Con	After overhearin	42.990906	[{"name": "New I	[{"iso_3166_1": '	2007-12-04	372234864	113	[{"isc
26	207000000	[{"id": 12, "name": "Adventure"}, {"i		254	[{"id": 774, "nam	en	King Kong	In 1933 New Yor	61.22601	[{"name": "Wingf	[{"iso_3166_1": '	2005-12-14	550000000	187	[{"isc

	A	B	C	
1	movie_id	title	cast	crew
2	19995	Avatar	[{"cast_id": 242,	[{"credit_id": "52fe48009
3	285	Pirates of the Ca	[{"cast_id": 4, "ch	[{"credit_id": "52fe4232c
4	206647	Spectre	[{"cast_id": 1, "ch	[{"credit_id": "54805967c
5	49026	The Dark Knight	[{"cast_id": 2, "ch	[{"credit_id": "52fe4781c
6	49529	John Carter	[{"cast_id": 5, "ch	[{"credit_id": "52fe479ac
7	559	Spider-Man 3	[{"cast_id": 30, "c	[{"credit_id": "52fe4252c
8	38757	Tangled	[{"cast_id": 34, "c	[{"credit_id": "52fe46db9
9	99861	Avengers: Age o	[{"cast_id": 76, "c	[{"credit_id": "55d5f7d4c
10	767	Harry Potter and	[{"cast_id": 3, "ch	[{"credit_id": "52fe4273c
11	209112	Batman v Super	[{"cast_id": 18, "c	[{"credit_id": "553bf2369
12	1452	Superman Retur	[{"cast_id": 3, "ch	[{"credit_id": "553bef6a9
13	10764	Quantum of Sola	[{"cast_id": 1, "ch	[{"credit_id": "52fe43b29
14	58	Pirates of the Ca	[{"cast_id": 37, "c	[{"credit_id": "52fe4211c
15	57201	The Lone Range	[{"cast_id": 4, "ch	[{"credit_id": "52fe4928c
16	49521	Man of Steel	[{"cast_id": 2, "ch	[{"credit_id": "52fe4799c
17	2454	The Chronicles o	[{"cast_id": 1, "ch	[{"credit_id": "55a239e69
18	24428	The Avengers	[{"cast_id": 46, "c	[{"credit_id": "52fe4495c
19	1865	Pirates of the Ca	[{"cast_id": 15, "c	[{"credit_id": "566b4f54c
20	41154	Men in Black 3	[{"cast_id": 4, "ch	[{"credit_id": "52fe45b7c
21	122917	The Hobbit: The	[{"cast_id": 10, "c	[{"credit_id": "548ad49a9
22	1930	The Amazing Sp	[{"cast_id": 56, "c	[{"credit_id": "5395a60dc
23	20662	Robin Hood	[{"cast_id": 1, "ch	[{"credit_id": "52fe43f2c3
24	57158	The Hobbit: The	[{"cast_id": 3, "ch	[{"credit_id": "52fe4926c
25	2268	The Golden Con	[{"cast_id": 43, "c	[{"credit_id": "52fe4348c
26	254	King Kong	[{"cast_id": 5, "ch	[{"credit_id": "52fe422ec

By taking the vote_count and vote_average into account , calculated the weighted_rating by using the IMDB formula

```
[ ]  def weighted_rating(x, m=m, C=C):
         v = x['vote_count']
         R = x['vote_average']
         return (v/(v+m) * R) + (m/(m+v) * C)
```

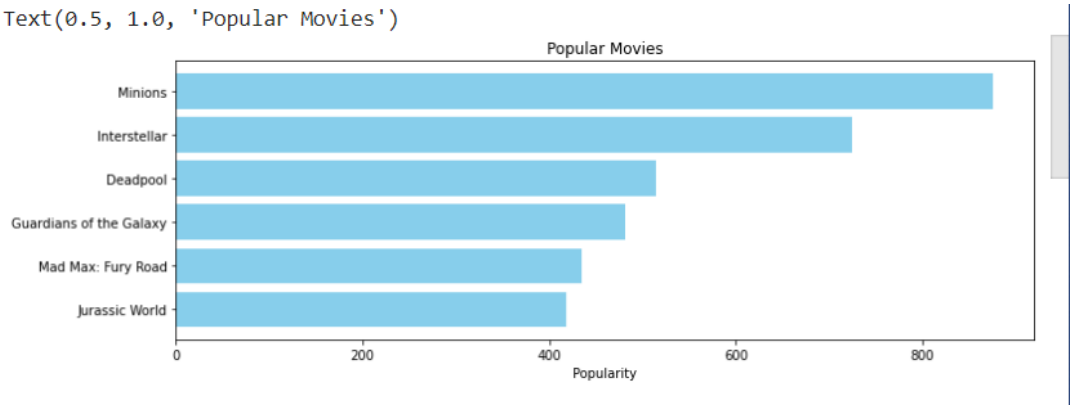
Created a new feature score and calculated the weighted rating value . Sorted the movies on scores and printed the top movies

```
[ ]  q_movies = q_movies.sort_values('score', ascending=False)

     q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

BAR CHART:

```
Text(0.5, 1.0, 'Popular Movies')
```



Preprocessing involves removal of stop words like ‘the’ , ‘a’. Replaced NaN with empty string.

```
[ ] from sklearn.feature_extraction.text import TfidfVectorizer
    tfidf = TfidfVectorizer(stop_words='english')


    df2['overview'] = df2['overview'].fillna('')
    tfidf_matrix = tfidf.fit_transform(df2['overview'])


    tfidf_matrix.shape
```

```
(4803, 20978)
```

Created a function that takes in movie genres and matches with the scores .The cosine similarity matches the similar movies and displays the top movies based on the scores that are higher.

```
get_recommendations('The Dark Knight Rises')
```

```
65                              The Dark Knight
299                              Batman Forever
428                              Batman Returns
1359                                    Batman
3854    Batman: The Dark Knight Returns, Part 2
119                               Batman Begins
2507                                  Slow Burn
9            Batman v Superman: Dawn of Justice
1181                                       JFK
210                              Batman & Robin
Name: title, dtype: object
```

DATASET 2 - SMALL DATASET

A	B	C
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller
11	American President, The (1995)	Comedy|Drama|Romance
12	Dracula: Dead and Loving It (1995)	Comedy|Horror
13	Balto (1995)	Adventure|Animation|Children
14	Nixon (1995)	Drama
15	Cutthroat Island (1995)	Action|Adventure|Romance
16	Casino (1995)	Crime|Drama
17	Sense and Sensibility (1995)	Drama|Romance
18	Four Rooms (1995)	Comedy
19	Ace Ventura: When Nature Calls (199	Comedy
20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
21	Get Shorty (1995)	Comedy|Crime|Thriller

Since some movies don't have the information about the year in the column , we need to handle them . I have changed the column name from title to title year , removed leading and ending whitespaces in the title_year and created separate columns for title and year.

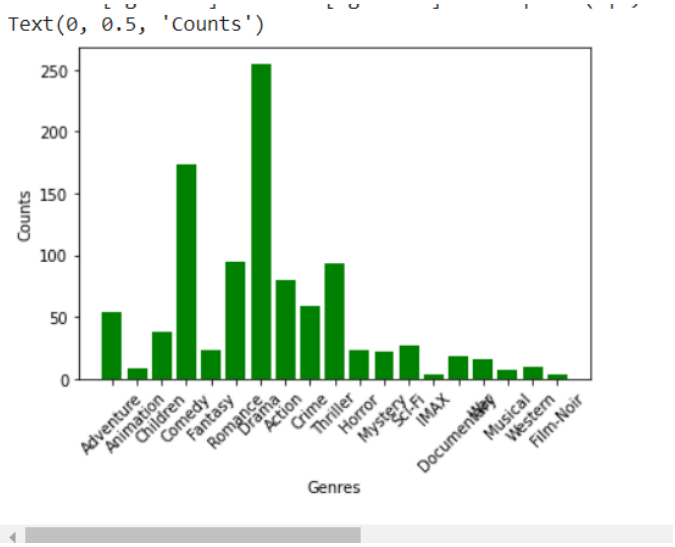
```
def extract_title(title):
    year = title[len(title)-5:len(title)-1]


    if year.isnumeric():
        title_no_year = title[:len(title)-7]
        return title_no_year
    else:
        return title
def extract_year(title):
    year = title[len(title)-5:len(title)-1]
    if year.isnumeric():
        return int(year)
    else:
        return np.nan
movies.rename(columns={'title':'title_year'}, inplace=True)
movies['title_year'] = movies['title_year'].apply(lambda x: x.strip())
movies['title'] = movies['title_year'].apply(extract_title)
movies['year'] = movies['title_year'].apply(extract_year)
```

Removed the movies without genre info and resetted the index , Then removed the ‘|’ in the genre column and counted the number of occurrence for each genre in dataset

```
movies['genres'] = movies['genres'].str.replace('|',' ')

counts = dict()
for i in movies.index:
   for g in movies.loc[i,'genres'].split(' '):
      if g not in counts:
         counts[g] = 1
      else:
         counts[g] = counts[g] + 1
```

BARCHART :

```
Text(0, 0.5, 'Counts')
```



By using `TfidfVectorizer, each feature index in the column is given with a unique token id .` Builded the cosine similarity matrix and found the similarity between them.

```
print(list(enumerate(tfidf_vector.get_feature_names())))
```

```
[(0, 'action'), (1, 'adventure'), (2, 'animation'), (3, 'children'), (4, 'comedy'),
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarnin
  warnings.warn(msg, category=FutureWarning)
```

```
print(sim_matrix)
```

```
[[1.         0.75572094 0.15025974 ... 0.09574404 0.         0.24605568]
 [0.75572094 1.         0.         ... 0.         0.         0.        ]
 [0.15025974 0.         1.         ... 0.23762251 0.67115372 0.61067374]
 ...
 [0.09574404 0.         0.23762251 ... 1.         0.         0.38911532]
 [0.         0.         0.67115372 ... 0.         1.         0.        ]
 [0.24605568 0.         0.61067374 ... 0.38911532 0.         1.        ]]
```

Fuzzywuzzy ,the similar matrices are grouped together . find_closest_title is used to return the most similar title to the word that is given as input.

```
def find_closest_title(title):
    leven_scores = list(enumerate(movies['title'].apply(matching_score, b=title)))
    sorted_leven_scores = sorted(leven_scores, key=lambda x: x[1], reverse=True)
    closest_title = get_title_from_index(sorted_leven_scores[0][0])
    distance_score = sorted_leven_scores[0][1]
    return closest_title, distance_score
```

Finally , the distance score is set to a threshold value of 100 , it even calculates correctly when the user makes a mistake in typing . And removed the typed movie itself . Then the list of movies is displayed and we can alter the no of input movies

```
contents_based_recommender('Monsters, Inc.', 20)
```

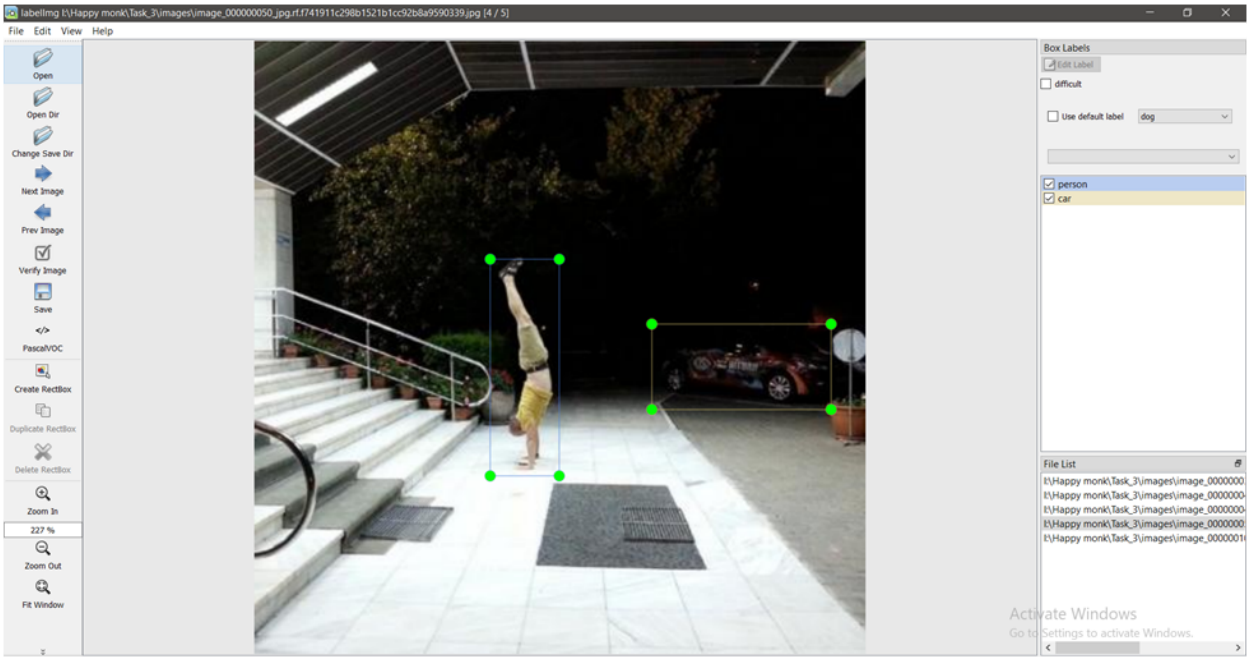
```
Did you mean Dunston Checks In?

Here's the list of movies similar to Dunston Checks In.

It Takes Two (1995)
Big Green, The (1995)
Heavyweights (Heavy Weights) (1995)
Richie Rich (1994)
Baby-Sitters Club, The (1995)
Now and Then (1995)
Babe (1995)
White Balloon, The (Badkonake sefid) (1995)
Fluke (1995)
Little Princess, A (1995)
Gordy (1995)
Flintstones, The (1994)
Addams Family Values (1993)
Mighty Morphin Power Rangers: The Movie (1995)
Tom and Huck (1995)
Amazing Panda Adventure, The (1995)
Casper (1995)
Far From Home: The Adventures of Yellow Dog (1995)
Lassie (1994)
Free Willy 2: The Adventure Home (1995)
```

TASK 2

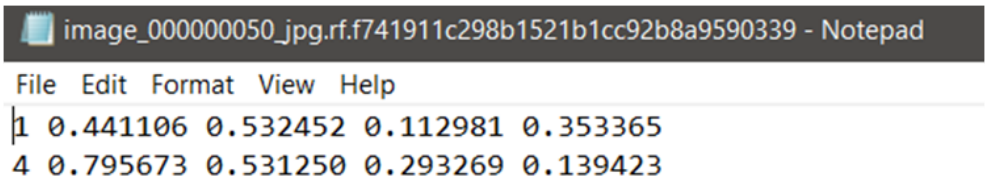
QUESTION 2

2. Collect data (images) and annotate them for two classes: Person and vehicle. You may use platforms such as LabelImg for annotations. You may limit to 800 images for the dataset. Perform object detection on your collected dataset and find the mean distance between the two classes in each image. You may use YOLOv5 for detection.

Annotating using LabelImg:



Image_name.xml file for yolo model:



Using Pre Trained YOLOV5 model - COCO model:

yolov5 prerequisites

Yolov5 Prerequisite

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 14461, done.
remote: Counting objects: 100% (90/90), done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 14461 (delta 56), reused 53 (delta 25), pack-reused 14371
Receiving objects: 100% (14461/14461), 13.57 MiB | 9.62 MiB/s, done.
Resolving deltas: 100% (9957/9957), done.
/kaggle/working/yolov5
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended
o use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

Importing the necessary libraries and fitting the pretrained model:

Import Libraries

```
import torch
from matplotlib import pyplot as plt
import numpy as np
import cv2
```

Loading Pretrained Model

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
model.classes = [0, 2]  # person and car
```

```
Downloading: "https://github.com/ultralytics/yolov5/archive/master.zip" to /root/.cache/torch/hub/master.zip
YOLOv5 🚀 2022-12-11 Python-3.7.12 torch-1.11.0+cpu CPU

Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
  0%|          | 0.00/14.1M [00:00, ?B/s]
Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
Adding AutoShape...
```

Object Detection on Sample Image (Car - Person):

Pretrained-Model on Sample Data

```
img = 'I:/car-person-v2-roboflow/Car-Person-v2-Roboflow-Owais-Ahmad/test/images/image_000000102_jpg.rf.79e39dcbcd8179ea9e837ce05fb096a1.jpg'

# Inference
results = model(img)
results.print()  # or .show(), .save()
```

```
image 1/1: 416x416 5 persons, 2 cars
Speed: 72.7ms pre-process, 360.1ms inference, 15.9ms NMS per image at shape (1, 3, 640, 640)
```

```
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
```

Calculating mean distance between classes:
The midpoints of the objects and mean difference of two points are calculated

Calculating Mean Difference

```
res = results.pandas().xyxy[0]
rn = {'car':'vehicle','person':'person'}
res['name'] = res['name'].map(rn)
```

```
def midpoint(xmin, ymin, xmax, ymax):
  center_w = xmax - xmin
  center_h = ymax - ymin
  center_x = 0.5*(xmin + xmax)
  center_y = 0.5*(ymin + ymax)
  return (center_x, center_y)
center = []
for i in range(4):
  center.append(midpoint(res['xmin'][i], res['ymin'][i], res['xmax'][i], res['ymax'][i]))
distance = pow(pow((center[0][0]-center[1][0]),2) + pow((center[0][1]-center[1][1]),2),0.5)
```

Plotting the difference

The distance between objects is plot in a form of red line

```
image = np.squeeze(results.render())
window_name = 'Image'
start_point = (int(center[0][0]), int(center[0][1]))

end_point = (int(center[1][0]), int(center[1][1]))

# Blue color in BGR
color = (255, 0, 0)
# Line thickness
thickness = 15
label = str(int(distance))
text_org = (int(center[0][0] + 200) , int(center[0][1]) - 50)
image = cv2.line(image, start_point, end_point, color, thickness)
image = cv2.putText(image, label, text_org, cv2.FONT_HERSHEY_SIMPLEX, 10, color, 12,  cv2.LINE_AA)
# Displaying the image
plt.imshow(image)
plt.show()
```

Inferences of Study

- Annotations helps to data to know the coordinates to bounding boxes and object detection
- The pretrained model is used to handle data with high speed and better accuracy.

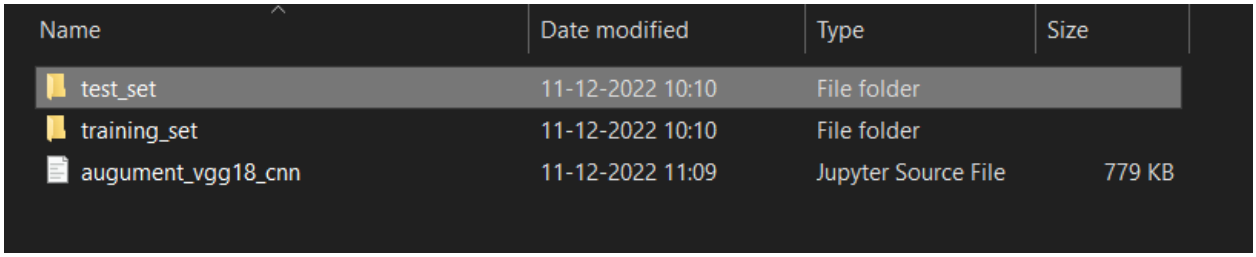
TASK 3

QUESTION 3

Download an image dataset of your choice for binary class classification. Perform the data augmentation techniques like flipping, rotation and transformation. Apply at least two object classification techniques both on the augmented as well as on the original dataset. Display the performance of the Algorithms. Prepare a comparison chart.

DATASET:

We have used cats and dogs dataset for this binary classification. Both train and testset consists of dogs and cats images.



Models Used:

- Baseline Convolutional Neural Networks
- Pretrained Model - VGG-16

BaseLine CNN:

Data Augmentation

```
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   zoom_range = 0.2,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   vertical_flip = True,
                                   fill_mode = 'nearest',
                                   validation_split = 0.1)
```

:

CNN Model Fitting:

```
cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding="same", activation="relu", input_shape=[256,256,
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2, padding='valid'))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding="same", activation="relu"))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2, padding='valid'))
cnn.add(tf.keras.layers.Flatten())
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Python

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

Python

```
cnn.summary()
```

Python

VGG-16 Model Fitting:

```
base_model = VGG16(input_shape = IMG_SHAPE, include_top=False, weights='imagenet')
```

```
base_model.trainable = False
```

```
inputs = Input(shape = IMG_SHAPE)

x = base_model(inputs, training = False)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
outputs = Dense(1, activation='sigmoid')(x)

model = Model(inputs = inputs, outputs = outputs)
```

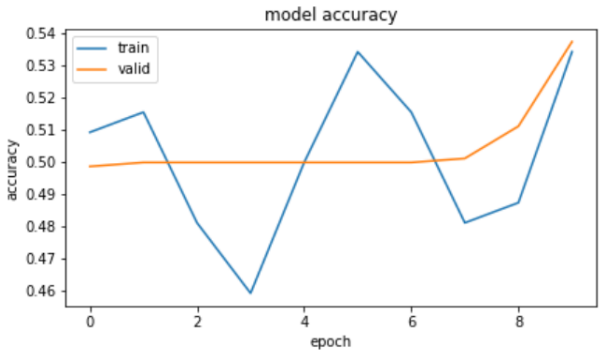
```
base_learning_rate = 0.001
optimizer = Adam(learning_rate=base_learning_rate)
```

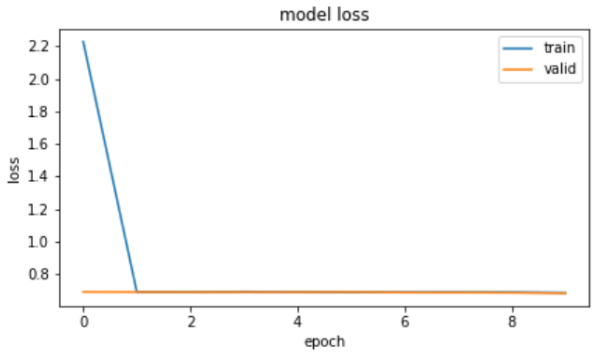
```
model.compile(
    optimizer = optimizer,
    loss = BinaryCrossentropy(),
    metrics = ['accuracy']
)
```

Results:

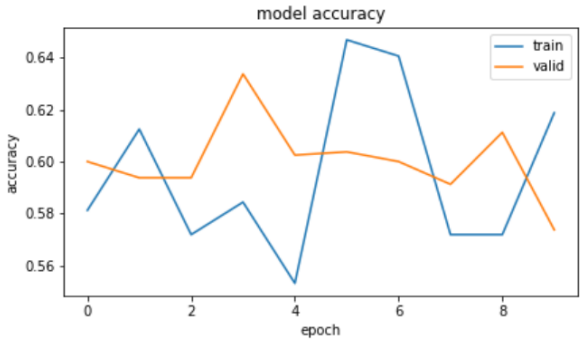
BaseLine CNN

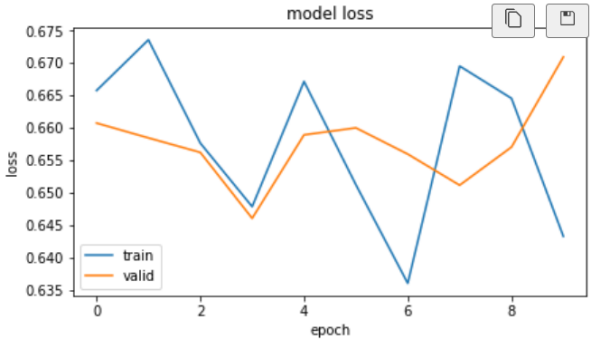
Without Augmentation:





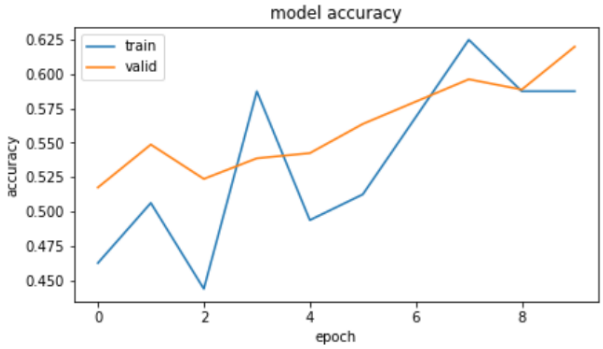
With Augmentation:

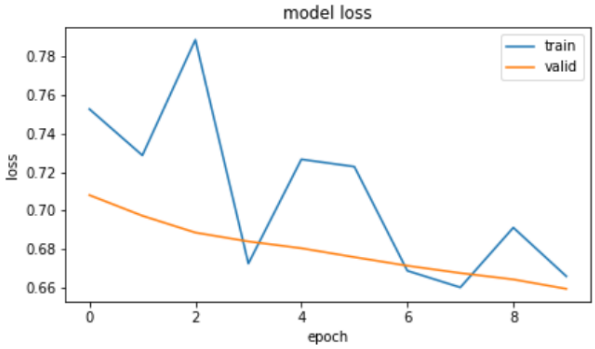




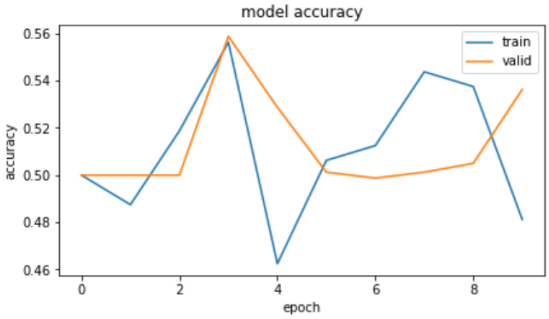
VGG-18

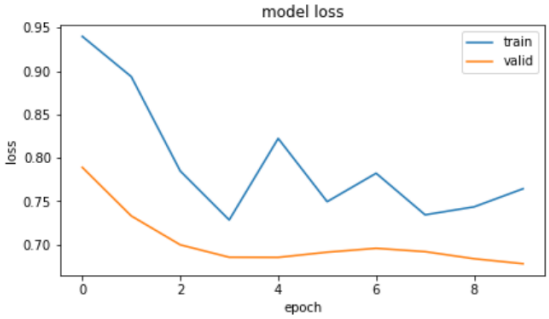
Without Augmentation:





With Augmentation:





Inferences of Study

- Augmentation Data for both baseline model and vgg-16 performs better than Non-Augmented Data.
- Pretrained Model along with augmented data performs better than baseline models.

TASK 4

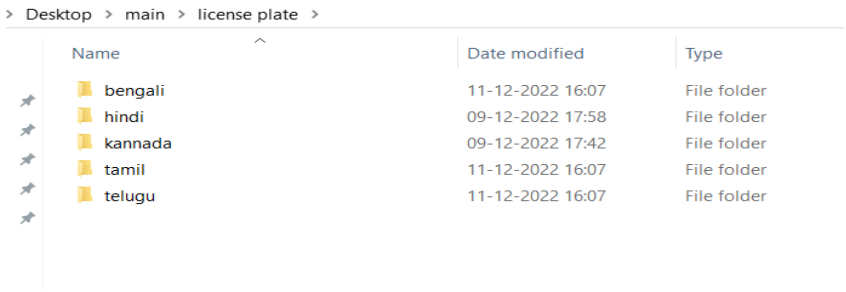
QUESTION 4:

4. Collect images of vehicles with license plates written in Indian regional languages (eg. Hindi, Kannada, Tamil, Telugu, Bengali, etc.). Apply Image augmentation techniques on the collected images. Maintain separate folders for different language license plates. You may limit to 800 images in the dataset including the augmented images.

The aim is to get the license plates of different Indian languages and augment them and map them to the respective folders.

DATASET:

It consists of different license plates taken from different sites . I took 7 images for each class .Total of 35 images. It is saved in the JPEG format . It is stored separately in each folder.



Sdir gives the original path , augdir is the place to store the files. N - no of augmented images . I have given some additional features like img_size gives the (height and width) , color__mode is set to rgb by default. save _prefix is the prefix for the augmented images that will be saved for the new images .

Type of augmentation that is used on images

```
gen = ImageDataGenerator(rescale=1./255, rotation_range =15,
                         fill_mode = 'nearest', shear_range=0.2,
                         zoom_range=0.2, width_shift_range = 0.2,
                         height_shift_range = 0.2, horizontal_flip = True,
                         brightness_range=[0.5, 1.5])
```

Starting with an empty directory and creating a new one if the old path does not exist. If it already exists with some images it will delete them and create a new one , that prevents appending in the same files.

```
df=df.copy()
if os.path.isdir(augdir)
    shutil.rmtree(augdir)
os.mkdir(augdir)
for label in df['labels'].unique():
    classpath=os.path.join(augdir,label)
    os.mkdir(classpath)
total=0
```

Specifying no of images need to be created in the each folder and gives the total no of augmented images in each class.

```
In [5]: sdir=r'C:/Users/DELL/Desktop/main/license plate'
df=make_dataframe(sdir)
print (df.head())
print ('length of dataframe is ',len(df))

augdir=r'C:/Users/DELL/Desktop/main/new'
n=7
img_size=(224,224)
make_and_store_images(df, augdir, n,  img_size,  color_mode='rgb', save_prefix='aug-',save_format='jpg')
```

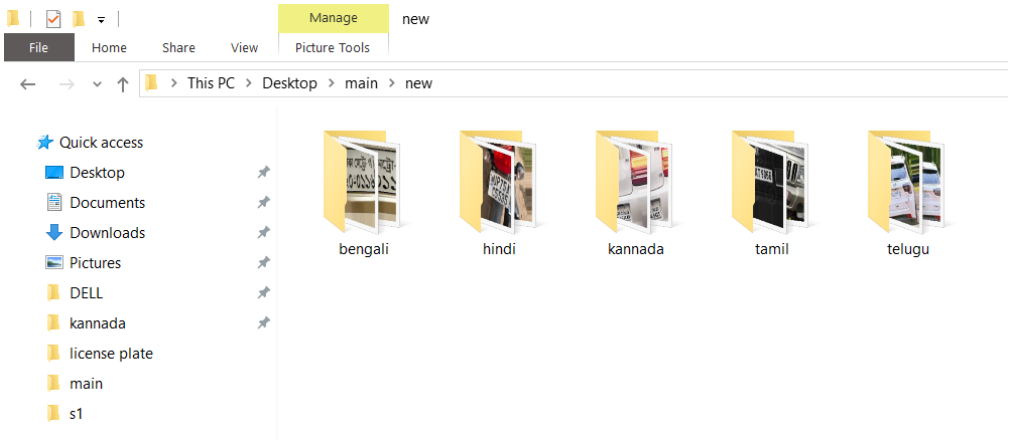
```
bengali                 : 100%|| 9/9 [00:00<00:00, 9022.16fil
es/s]
hindi                   : 100%|| 9/9 [00:00<00:00, 9418.35fil
es/s]
kannada                 : 100%|| 7/7 [00:00<00:00, 6983.86fil
es/s]
tamil                   : 100%|| 12/12 [00:00<?, ?fil
es/s]
telugu                  : 100%|| 7/7 [00:00<00:00, 17783.24fil
es/s]
                                        filepaths   labels
0  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
1  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
2  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
3  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
4  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
length of dataframe is  44
```

Finally specify the path where to take the images and the path where to store and name of the file to be created . Here I have given new , it will create a folder name new and itself create the subfolders by taking the name form sdir path.

```
make_and_store_images(df, augdir, n,  img_size,  color_mode='rgb', save_prefix='aug-',save_format='jpg')
```

```
es/s]
telugu                  : 100%|| 7/7 [00:00<?, ?fil
es/s]
                                        filepaths   labels
0  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
1  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
2  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
3  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
4  C:/Users/DELL/Desktop/main/license plate\benga...  bengali
length of dataframe is  35
Found 7 validated image filenames.      for class           bengali            creating  14   augmented images

Found 7 validated image filenames.      for class            hindi             creating  14   augmented images
Found 7 validated image filenames.      for class           kannada            creating  14   augmented images
Found 7 validated image filenames.      for class            tamil             creating  14   augmented images
Found 7 validated image filenames.      for class            telugu            creating  14   augmented images
Total Augmented images created=  70
```

OUTPUT:



BENGALI :

KANNADA:

HINDI:

TAMIL:


aug-_0_854094


aug-_0_3732803


aug-_1_1188651

aug-_1_6317637

aug-_2_7832624

aug-_2_9082668

aug-_3_2442602

aug-_3_9884316


aug-_4_2125658


aug-_4_3317186


aug-_5_3335937


aug-_5_5426546


aug-_6_369156


aug-_6_1024114

TELUGU:


aug-_0_3180780


aug-_0_4689435

aug-_1_1753231

aug-_1_3825342

aug-_2_6512090

aug-_2_8336225


aug-_3_3703468


aug-_3_8227960


aug-_4_1866967


aug-_4_8434399


aug-_5_1103630


aug-_5_3722346


aug-_6_1553889


aug-_6_5303885

TASK 5

QUESTION

Download the pytorch check point file from here (link to be added) and convert the file to .onnx

- Perform inferences on an onnx runtime session.
- Write a wrapper to perform the inference on video feed from webcam.
- Relevant papers to the .pth file:

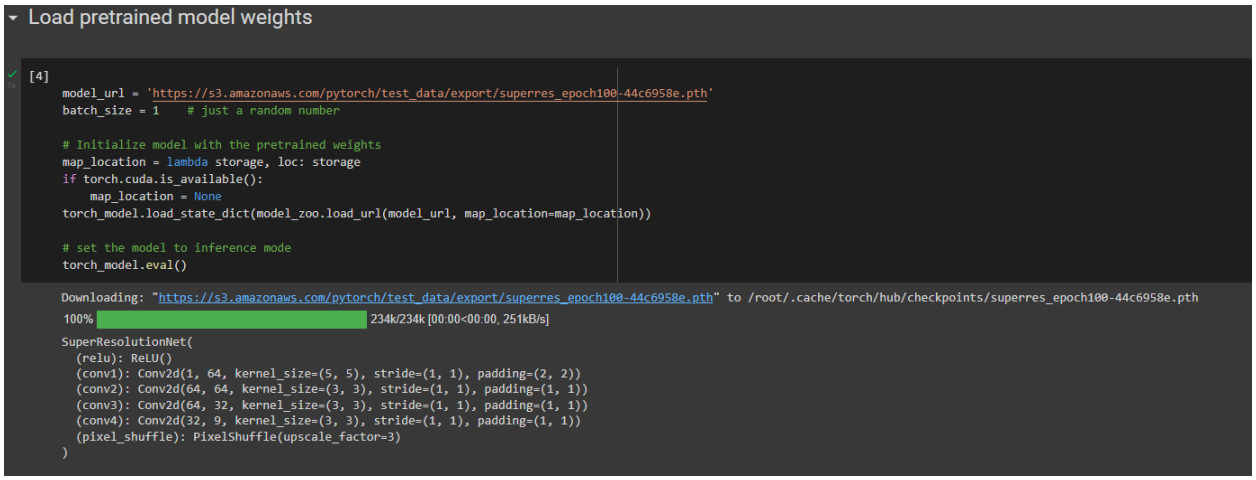
Exporting a Model from PyTorch to ONNX and Running it using ONNX Runtime

Super-resolution model in PyTorch:

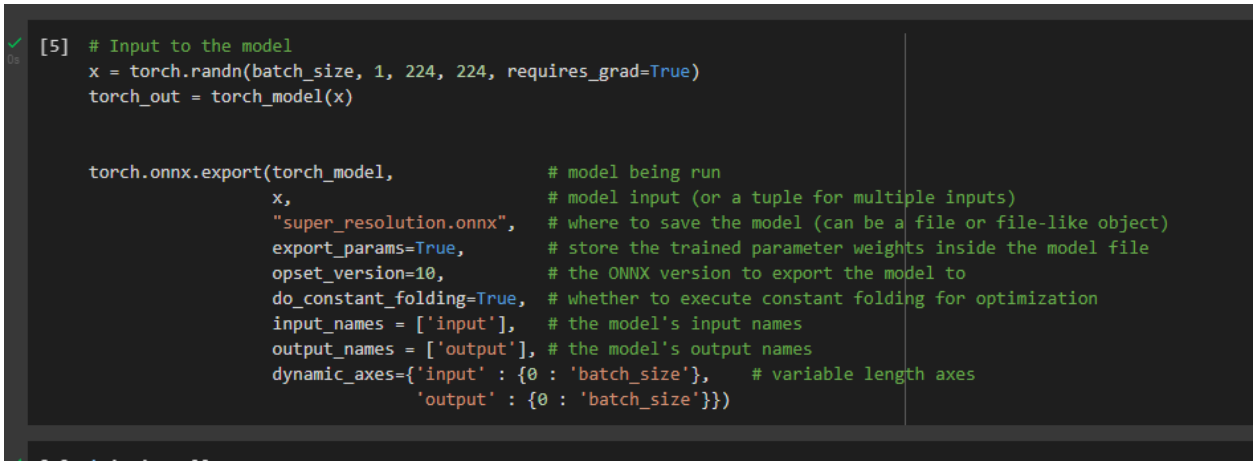
Super-resolution is a way of increasing the resolution of images, videos and is widely used in image processing or video editing.

```python
# Super Resolution model definition in PyTorch
import torch.nn as nn
import torch.nn.init as init


class SuperResolutionNet(nn.Module):
    def __init__(self, upscale_factor, inplace=False):
        super(SuperResolutionNet, self).__init__()

        self.relu = nn.ReLU(inplace=inplace)
        self.conv1 = nn.Conv2d(1, 64, (5, 5), (1, 1), (2, 2))
        self.conv2 = nn.Conv2d(64, 64, (3, 3), (1, 1), (1, 1))
        self.conv3 = nn.Conv2d(64, 32, (3, 3), (1, 1), (1, 1))
        self.conv4 = nn.Conv2d(32, upscale_factor ** 2, (3, 3), (1, 1), (1, 1))
        self.pixel_shuffle = nn.PixelShuffle(upscale_factor)

        self._initialize_weights()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.relu(self.conv3(x))
        x = self.pixel_shuffle(self.conv4(x))
        return x

    def _initialize_weights(self):
        init.orthogonal_(self.conv1.weight, init.calculate_gain('relu'))
        init.orthogonal_(self.conv2.weight, init.calculate_gain('relu'))
        init.orthogonal_(self.conv3.weight, init.calculate_gain('relu'))
        init.orthogonal_(self.conv4.weight)

# Create the super-resolution model by using the above model definition.
torch_model = SuperResolutionNet(upscale_factor=3)
```

Training Super-resolution Model using Pretrained-weight



Exporting a Model from PyTorch to ONNX:



Now let's compute the output using ONNX Runtime's Python APIs. This part can normally be done in a separate process or on another machine, but we will continue in the same process so that we can verify that ONNX Runtime and PyTorch are computing the same value for the network.

In order to run the model with ONNX Runtime, we need to create an inference session for the model with the chosen configuration parameters (here we use the default config). Once the session is created, we evaluate the model using the run() api. The output of this call is a list containing the outputs of the model computed by ONNX Runtime.

Inferences on an onnx runtime session.

Inferences on an onnx runtime session.

```
[11] import onnxruntime

     ort_session = onnxruntime.InferenceSession("super_resolution.onnx")

     def to_numpy(tensor):
         return tensor.detach().cpu().numpy() if tensor.requires_grad else tensor.cpu().numpy()

     # compute ONNX Runtime output prediction
     ort_inputs = {ort_session.get_inputs()[0].name: to_numpy(x)}
     ort_outs = ort_session.run(None, ort_inputs)

     # compare ONNX Runtime and PyTorch results
     np.testing.assert_allclose(to_numpy(torch_out), ort_outs[0], rtol=1e-03, atol=1e-05)

     print("Exported model has been tested with ONNXRuntime, and the result looks good!")

     Exported model has been tested with ONNXRuntime, and the result looks good!
```

Running the model on an image using ONNX Runtime

Running the model on an image using ONNX Runtime

```
[13] from PIL import Image
     import torchvision.transforms as transforms

     img = Image.open("A-Cat.jpg")

     resize = transforms.Resize([224, 224])
     img = resize(img)

     img_ycbcr = img.convert('YCbCr')
     img_y, img_cb, img_cr = img_ycbcr.split()

     to_tensor = transforms.ToTensor()
     img_y = to_tensor(img_y)
     img_y.unsqueeze_(0)

     tensor([[[[0.7725, 0.7843, 0.7804,  ..., 0.6824, 0.6784, 0.6902],
               [0.7882, 0.7843, 0.7725,  ..., 0.6549, 0.6667, 0.6824],
               [0.8039, 0.7961, 0.7843,  ..., 0.6588, 0.6706, 0.6784],
               ...,
               [0.7333, 0.7294, 0.7255,  ..., 0.1020, 0.0941, 0.0902],
               [0.7333, 0.7333, 0.7333,  ..., 0.1020, 0.0902, 0.0863],
               [0.7333, 0.7373, 0.7333,  ..., 0.0980, 0.0863, 0.0824]]]])

[19] ort_inputs = {ort_session.get_inputs()[0].name: to_numpy(img_y)}
     ort_outs = ort_session.run(None, ort_inputs)
     img_out_y = ort_outs[0]
```

Comparison between input and output data:

Comparision between input and output

```
display(final_img) #final_img
```

```
[21] display(img) #input image
```