**Madhav Institute of Technology and Science, Gwalior**

(Deemed to be University)

NAAC Accredited with A++ Grade

**Centre for Artificial Intelligence**



## Skill Based Mini Project Of

**Data Mining & Warehousing**

**(280601)**

**SUBMITTED BY**

Anand Bhadoriya (0901AM211010)

Aryan Singh (0901AM211015)

Neetesh Jatav (0901AM211034)

6th semester

Artificial Intelligence & Machine Learning (AIML)

JANUARY – JUNE, 2024

**SUBMITTED TO**

Dr. Shubha Mishra

Prof. Gaurisha Sisodiya

**ACKNOWLEDGEMENT**

I would like to express my greatest appreciation to all the individuals who have helped and supported me throughout this lab file. I am thankful to whole Centre for Artificial Intelligence department for their ongoing support during the experiments, from initial advice and provision of contact in the first stages through ongoing advice and encouragement, which led to the finals report of this lab file.

A special acknowledgement goes to my colleagues who help me in completing the file and by exchanging interesting ideas to deal with problems and sharing the experience.

I wish to thank our professor **Dr. Shubha Mishra** as well for her undivided support and interests which inspired me and encouraged me to go my own way without whom I would be unable to complete my project.

At the end, I want to thank my friends who displayed appreciation to my work and motivated me to continue my work.

Anand Bhadoriya (0901AM211010)

Aryan Singh (0901AM211015)

Neetesh Jatav (0901AM211034)

**MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE GWALIOR**

(A Govt. Aided UGC Autonomous Institute Affiliated to RGPV, Bhopal)

***NAAC Accredited with A++ Grade***

**CERTIFICATE**

This is certified that **Anand Bhadoriya (0901AM211010), Aryan Singh (0901AM211015) & Neetesh Jatav (0901AM211034)** has submitted the project report titled **House Price Prediction** under the mentorship of **Dr. Shubha Mishra & Prof. Gaurisha Sisodiya** in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in **Artificial Intelligence and Machine Learning** from Madhav Institute of Technology and Science, Gwalior.

**Dr. Shubha Mishra Prof. Gaurisha Sisodiya**

Faculty Mentor Faculty Mentor

Centre for Artificial Intelligence Centre for Artificial Intelligence

**Table of Content**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Type of Project** | **Page No.** | **Date** | **Signature** |
| **1.** | Micro Project |  |  |  |
| **2.** | Macro Project |  |  |  |
| **3.** | Mini Project |  |  |  |

# Micro Project

## Micro Project

* **Aim –** The project aims to analyze the dataset used for the House Price Prediction and perform exploratory data analysis on it.
* **Theory –**
  + **Dataset Analysis**

The project's dataset is curated using the House Price Prediction dataset from Kaggle, comprising 6347 rows and 19 Columns. This dataset serves as a foundational element for training the model, ensuring a diverse and representative collection of Various Areas.

###### Error Level Analysis (ELA)

Exploratory Data Analysis, or EDA, is the detective work of data science. Before diving into complex models, EDA involves a range of techniques to understand what your data is telling you. It's like examining a crime scene – uncovering patterns, trends, and potential suspects (outliers or errors).

By visualizing the data through charts and graphs, EDA helps you see hidden relationships between variables. You might discover unexpected connections or confirm existing hunches. This initial exploration sets the stage for further analysis. Imagine a map – EDA helps you navigate the landscape of your data, highlighting areas of interest for deeper investigation.

Ultimately, EDA is a conversation with your data. It allows you to ask questions, uncover its secrets, and ultimately gain valuable insights before drawing any conclusions.

Traffic Sign Detection

#### Micro Project

In [1]:

import numpy as np

import pandas as pd

import matplotlib as mat

import matplotlib.pyplot as plt

import sklearn as sk

from sklearn import metrics

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

import random

import seaborn as sb

import math

import cv2

from google.colab import drive

drive.mount('/content/drive')

In [2]:

**from** PIL **import** Image

**import** os

**from** pylab **import \* import** re

**from** PIL **import** Image, ImageChops, ImageEnhance

In [3]:

**def** get\_imlist(path):

**return** [os**.**path**.**join(path,f) **for** f **in** os**.**listdir(path) **if** f**.**endswith('.jpg')

In [10]:

**def** convert\_to\_ela\_image(path, quality): filename **=** path

resaved\_filename **=** filename**.**split('.')[0] **+** '\_resaved.jpg' ELA\_filename **=** filename**.**split('.')[0] **+** '\_ela.png'

im **=** Image**.**open(filename)**.**convert('RGB')

im**.**save(resaved\_filename, 'JPEG', quality**=**quality) resaved\_im **=** Image**.**open(resaved\_filename)

ela\_im **=** ImageChops**.**difference(im, resaved\_im) extrema **=** ela\_im**.**getextrema()

max\_diff **=** max([ex[1] **for** ex **in** extrema])

**if** max\_diff **==** 0: max\_diff **=** 1

scale **=** 255.0 **/** max\_diff

ela\_im **=** ImageEnhance**.**Brightness(ela\_im)**.**enhance(scale)

**return** ela\_im

### Sample: Real Image

In [5]:

Image**.**open('datasets/Au/Au\_ani\_00002.jpg')

Out[5]:

In [6]:

convert\_to\_ela\_image('datasets/Au/Au\_ani\_00002.jpg',90)

Out[6]:

### Sample: Fake Image

In [7]:

Image**.**open('datasets/Tp/Tp\_D\_NRN\_S\_N\_ani10171\_ani00001\_12458.jpg')

Out[7]:

In [8]:

convert\_to\_ela\_image('datasets/Tp/Tp\_D\_NRN\_S\_N\_ani10171\_ani00001\_12458.jpg', 90)

Out[8]:

# Macro Project

## Macro Project

* **Aim –** The project aims to implement the dataset and deriving useful insights from it by using YOLOv8 model and then pre-processing the data.
* **Theory –**

**YOLOv8** - YOLO (You Only Look Once) is a popular family of object detection algorithms known for their real-time processing capabilities. YOLOv8 is the latest iteration, likely boasting improvements in accuracy, speed, or both over previous versions.

A pre-trained YOLOv8 model comes with weights already trained on a large dataset of images and corresponding bounding boxes for various objects. This pre-training provides a strong foundation for object detection tasks.

Here are some potential benefits of using a pre-trained YOLOv8 model:

* Faster Training: By leveraging the pre-trained weights, you can fine-tune the model on your specific dataset with less training time compared to training from scratch.
* Improved Accuracy: The pre-trained model has already learned generic features of objects, potentially improving its ability to detect objects in your new data.

To use a pre-trained YOLOv8 model, you'll typically need to:

1. Choose a pre-trained model variant: Different variants might be optimized for speed or accuracy depending on your needs.
2. Fine-tune the model: Adjust the model's final layers on your specific dataset to improve detection of your target objects.
3. Use the model for detection: Once fine-tuned, you can use the model to detect objects in new images or video streams.
   * **Dataset Preprocessing**

**Data Cleaning**: The dataset is subjected to cleaning processes to remove any corrupt, irrelevant, or duplicate data. This ensures that the dataset is free from noise or inconsistencies that could adversely affect model training.

**Image Augmentation**: Augmentation techniques are utilized to artificially increase the size and diversity of the dataset. This includes operations such as rotation, flipping, cropping, and zooming, which help expose the model to a wider variety

of image variations.

**Feature Extraction**: Relevant features are extracted from the images to capture essential information. Techniques such as edge detection, color histogram computation, and texture analysis may be employed to extract meaningful features from the images.

**Normalization:** Normalization techniques are applied to standardize the pixel values of images. This process ensures that all input features have a similar scale, preventing certain features from dominating the learning process.

**Dimensionality Reduction:** In cases where the feature space is large, dimensionality reduction techniques like Principal Component Analysis (PCA) or feature selection methods may be applied to reduce the computational complexity and improve model performance.

|  |  |
| --- | --- |
|  | Macro Project |
| Data Implementation using YOLOv8  *# Install Essential Libraries*  !pip install ultralytics  unfold\_moreShow hidden output  In [2]:  *# Import Essential Libraries*  import os  import random  import pandas as pd  from PIL import Image  import cv2  from ultralytics import YOLO  from IPython.display import Video  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  sns.set(style='darkgrid')  import pathlib  import glob  from tqdm.notebook import trange, tqdm  import warnings  warnings.filterwarnings('ignore')  In [3]:  Image\_dir = '/kaggle/input/cardetection/train/images'  num\_samples = 9  image\_files = os.listdir(Image\_dir)  *# Randomly select num\_samples images*  rand\_images = random.sample(image\_files, num\_samples)  fig, axes = plt.subplots(3, 3, figsize=(11, 11))  for i **in** range(num\_samples):  image = rand\_images[i]  ax = axes[i // 3, i % 3]  ax.imshow(plt.imread(os.path.join(Image\_dir, image)))  ax.set\_title(f'Image **{**i+1**}**')  ax.axis('off')  plt.tight\_layout()  plt.show()    In [ ]:    **3.1.2. Get Shape Of An Image For Using In Training Step**  In [4]:  *# Get the size of the image*  image = cv2.imread("/kaggle/input/cardetection/train/images/00000\_00000\_00012\_png.rf.23f94508dba03ef2f8bd187da2ec9c26.jpg")  h, w, c = image.shape  print(f"The image has dimensions **{**w**}**x**{**h**}** and **{**c**}** channels.")  The image has dimensions 416x416 and 3 channels. Try Pre-trained YOLOv8 For Detect Traffic Signs In [5]:  *# Use a pretrained YOLOv8n model*  model = YOLO("yolov8n.pt")  *# Use the model to detect object*  image = "/kaggle/input/cardetection/train/images/FisheyeCamera\_1\_00228\_png.rf.e7c43ee9b922f7b2327b8a00ccf46a4c.jpg"  result\_predict = model.predict(source = image, imgsz=(416))  *# show results*  plot = result\_predict[0].plot()  plot = cv2.cvtColor(plot, cv2.COLOR\_BGR2RGB)  display(Image.fromarray(plot))  Downloading https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8n.pt to 'yolov8n.pt'...  100%|██████████| 6.23M/6.23M [00:00<00:00, 75.8MB/s]  image 1/1 /kaggle/input/cardetection/train/images/FisheyeCamera\_1\_00228\_png.rf.e7c43ee9b922f7b2327b8a00ccf46a4c.jpg: 416x416 1 traffic light, 8.2ms  Speed: 6.0ms preprocess, 8.2ms inference, 587.7ms postprocess per image at shape (1, 3, 416, 416)    **Training Step**  In [6]:  *# Build from YAML and transfer weights*  Final\_model = YOLO('yolov8n.yaml').load('yolov8n.pt')  *# Training The Final Model*  Result\_Final\_model = Final\_model.train(data="/kaggle/input/cardetection/data.yaml",epochs=100, imgsz = 416, batch = 64 ,lr0=0.0001, dropout= 0.15, device = 0)  Transferred 355/355 items from pretrained weights  Ultralytics YOLOv8.1.25 🚀 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15102MiB)  **engine/trainer:** task=detect, mode=train, model=yolov8n.yaml, data=/kaggle/input/cardetection/data.yaml, epochs=100, time=None, patience=100, batch=64, imgsz=416, save=True, save\_period=-1, cache=False, device=0, workers=8, project=None, name=train, exist\_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single\_cls=False, rect=False, cos\_lr=False, close\_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi\_scale=False, overlap\_mask=True, mask\_ratio=4, dropout=0.15, val=True, split=val, save\_json=False, save\_hybrid=False, conf=None, iou=0.7, max\_det=300, half=False, dnn=False, plots=True, source=None, vid\_stride=1, stream\_buffer=False, visualize=False, augment=False, agnostic\_nms=False, classes=None, retina\_masks=False, embed=None, show=False, save\_frames=False, save\_txt=False, save\_conf=False, save\_crop=False, show\_labels=True, show\_conf=True, show\_boxes=True, line\_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=None, workspace=4, nms=False, lr0=0.0001, lrf=0.01, momentum=0.937, weight\_decay=0.0005, warmup\_epochs=3.0, warmup\_momentum=0.8, warmup\_bias\_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label\_smoothing=0.0, nbs=64, hsv\_h=0.015, hsv\_s=0.7, hsv\_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy\_paste=0.0, auto\_augment=randaugment, erasing=0.4, crop\_fraction=1.0, cfg=None, tracker=botsort.yaml, save\_dir=runs/detect/train  Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...  100%|██████████| 755k/755k [00:00<00:00, 14.6MB/s]  2024-03-10 14:29:24,893 INFO util.py:124 -- Outdated packages:  ipywidgets==7.7.1 found, needs ipywidgets>=8  Run `pip install -U ipywidgets`, then restart the notebook server for rich notebook output.  2024-03-10 14:29:25,759 INFO util.py:124 -- Outdated packages:  ipywidgets==7.7.1 found, needs ipywidgets>=8  Run `pip install -U ipywidgets`, then restart the notebook server for rich notebook output.  2024-03-10 14:29:28.533749: E external/local\_xla/xla/stream\_executor/cuda/cuda\_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered  2024-03-10 14:29:28.533854: E external/local\_xla/xla/stream\_executor/cuda/cuda\_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered  2024-03-10 14:29:28.724702: E external/local\_xla/xla/stream\_executor/cuda/cuda\_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered  Overriding model.yaml nc=80 with nc=15  from n params module arguments  0 -1 1 464 ultralytics.nn.modules.conv.Conv [3, 16, 3, 2]  1 -1 1 4672 ultralytics.nn.modules.conv.Conv [16, 32, 3, 2]  2 -1 1 7360 ultralytics.nn.modules.block.C2f [32, 32, 1, True]  3 -1 1 18560 ultralytics.nn.modules.conv.Conv [32, 64, 3, 2]  4 -1 2 49664 ultralytics.nn.modules.block.C2f [64, 64, 2, True]  5 -1 1 73984 ultralytics.nn.modules.conv.Conv [64, 128, 3, 2]  6 -1 2 197632 ultralytics.nn.modules.block.C2f [128, 128, 2, True]  7 -1 1 295424 ultralytics.nn.modules.conv.Conv [128, 256, 3, 2]  8 -1 1 460288 ultralytics.nn.modules.block.C2f [256, 256, 1, True]  9 -1 1 164608 ultralytics.nn.modules.block.SPPF [256, 256, 5]  10 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']  11 [-1, 6] 1 0 ultralytics.nn.modules.conv.Concat [1]  12 -1 1 148224 ultralytics.nn.modules.block.C2f [384, 128, 1]  13 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']  14 [-1, 4] 1 0 ultralytics.nn.modules.conv.Concat [1]  15 -1 1 37248 ultralytics.nn.modules.block.C2f [192, 64, 1]  16 -1 1 36992 ultralytics.nn.modules.conv.Conv [64, 64, 3, 2]  17 [-1, 12] 1 0 ultralytics.nn.modules.conv.Concat [1]  18 -1 1 123648 ultralytics.nn.modules.block.C2f [192, 128, 1]  19 -1 1 147712 ultralytics.nn.modules.conv.Conv [128, 128, 3, 2]  20 [-1, 9] 1 0 ultralytics.nn.modules.conv.Concat [1]  21 -1 1 493056 ultralytics.nn.modules.block.C2f [384, 256, 1]  22 [15, 18, 21] 1 754237 ultralytics.nn.modules.head.Detect [15, [64, 128, 256]]  YOLOv8n summary: 225 layers, 3013773 parameters, 3013757 gradients, 8.2 GFLOPs  Transferred 319/355 items from pretrained weights  **TensorBoard:** Start with 'tensorboard --logdir runs/detect/train', view at http://localhost:6006/  ---------------------------------------------------------------------------  UsageError Traceback (most recent call last)  Cell In[6], line 5  **2** Final\_model = YOLO('yolov8n.yaml').load('yolov8n.pt')  **4** # Training The Final Model  ----> 5 Result\_Final\_model = Final\_model.train(data="/kaggle/input/cardetection/data.yaml",epochs=100, imgsz = 416, batch = 64 ,lr0=0.0001, dropout= 0.15, device = 0)  File /opt/conda/lib/python3.10/site-packages/ultralytics/engine/model.py:654, in Model.train(self, trainer, \*\*kwargs)  **651** **pass**  **653** self.trainer.hub\_session = self.session # attach optional HUB session  --> 654 self.trainer.train()  **655** # Update model and cfg after training  **656** **if** RANK **in** (-1, 0):  File /opt/conda/lib/python3.10/site-packages/ultralytics/engine/trainer.py:213, in BaseTrainer.train(self)  **210** ddp\_cleanup(self, str(file))  **212** **else**:  --> 213 self.\_do\_train(world\_size)  File /opt/conda/lib/python3.10/site-packages/ultralytics/engine/trainer.py:327, in BaseTrainer.\_do\_train(self, world\_size)  **325** **if** world\_size > 1:  **326** self.\_setup\_ddp(world\_size)  --> 327 self.\_setup\_train(world\_size)  **329** nb = len(self.train\_loader) # number of batches  **330** nw = max(round(self.args.warmup\_epochs \* nb), 100) **if** self.args.warmup\_epochs > 0 **else** -1 # warmup iterations  File /opt/conda/lib/python3.10/site-packages/ultralytics/engine/trainer.py:240, in BaseTrainer.\_setup\_train(self, world\_size)  **237** """Builds dataloaders and optimizer on correct rank process."""  **239** # Model  --> 240 self.run\_callbacks("on\_pretrain\_routine\_start")  **241** ckpt = self.setup\_model()  **242** self.model = self.model.to(self.device)  File /opt/conda/lib/python3.10/site-packages/ultralytics/engine/trainer.py:176, in BaseTrainer.run\_callbacks(self, event)  **174** """Run all existing callbacks associated with a particular event."""  **175** **for** callback **in** self.callbacks.get(event, []):  --> 176 callback(self)  File /opt/conda/lib/python3.10/site-packages/ultralytics/utils/callbacks/wb.py:112, in on\_pretrain\_routine\_start(trainer)  **110** **def** on\_pretrain\_routine\_start(trainer):  **111** """Initiate and start project if module is present."""  --> 112 wb.run **or** wb.init(project=trainer.args.project **or** "YOLOv8", name=trainer.args.name, config=vars(trainer.args))  File /opt/conda/lib/python3.10/site-packages/wandb/sdk/wandb\_init.py:1195, in init(job\_type, dir, config, project, entity, reinit, tags, group, name, notes, magic, config\_exclude\_keys, config\_include\_keys, anonymous, mode, allow\_val\_change, resume, force, tensorboard, sync\_tensorboard, monitor\_gym, save\_code, id, settings)  **1193** **if** logger **is** **not** **None**:  **1194** logger.exception(str(e))  -> 1195 **raise** e  **1196** **except** **KeyboardInterrupt** **as** e:  **1197** **assert** logger  File /opt/conda/lib/python3.10/site-packages/wandb/sdk/wandb\_init.py:1172, in init(job\_type, dir, config, project, entity, reinit, tags, group, name, notes, magic, config\_exclude\_keys, config\_include\_keys, anonymous, mode, allow\_val\_change, resume, force, tensorboard, sync\_tensorboard, monitor\_gym, save\_code, id, settings)  **1170** **try**:  **1171** wi = \_WandbInit()  -> 1172 wi.setup(kwargs)  **1173** **assert** wi.settings  **1174** except\_exit = wi.settings.\_except\_exit  File /opt/conda/lib/python3.10/site-packages/wandb/sdk/wandb\_init.py:306, in \_WandbInit.setup(self, kwargs)  **303** settings.update(init\_settings, source=Source.INIT)  **305** **if** **not** settings.\_offline **and** **not** settings.\_noop:  --> 306 wandb\_login.\_login(  **307** anonymous=kwargs.pop("anonymous", **None**),  **308** force=kwargs.pop("force", **None**),  **309** \_disable\_warning=**True**,  **310** \_silent=settings.quiet **or** settings.silent,  **311** \_entity=kwargs.get("entity") **or** settings.entity,  **312** )  **314** # apply updated global state after login was handled  **315** wl = wandb.setup()  File /opt/conda/lib/python3.10/site-packages/wandb/sdk/wandb\_login.py:317, in \_login(anonymous, key, relogin, host, force, timeout, \_backend, \_silent, \_disable\_warning, \_entity)  **314** **return** logged\_in  **316** **if** **not** key:  --> 317 wlogin.prompt\_api\_key()  **319** # make sure login credentials get to the backend  **320** wlogin.propogate\_login()  File /opt/conda/lib/python3.10/site-packages/wandb/sdk/wandb\_login.py:247, in \_WandbLogin.prompt\_api\_key(self)  **241** **if** status == ApiKeyStatus.NOTTY:  **242** directive = (  **243** "wandb login [your\_api\_key]"  **244** **if** self.\_settings.\_cli\_only\_mode  **245** **else** "wandb.login(key=[your\_api\_key])"  **246** )  --> 247 **raise** UsageError("api\_key not configured (no-tty). call " + directive)  **249** self.update\_session(key, status=status)  **250** self.\_key = key  UsageError: api\_key not configured (no-tty). call wandb.login(key=[your\_api\_key])  **Validation Step**  In [ ]:  list\_of\_metrics = ["P\_curve.png","R\_curve.png","confusion\_matrix.png"]  In [ ]:  *# Load the image*  for i **in** list\_of\_metrics:  image = cv2.imread(f'/kaggle/working/runs/detect/train/**{**i**}**')  *# Create a larger figure*  plt.figure(figsize=(16, 12))  *# Display the image*  plt.imshow(image)  *# Show the plot*  plt.show()  In [ ]:  Result\_Final\_model = pd.read\_csv('/kaggle/working/runs/detect/train/results.csv')  Result\_Final\_model.tail(10)  In [ ]:  *# Read the results.csv file as a pandas dataframe*  Result\_Final\_model.columns = df.columns.str.strip()  *# Create subplots*  fig, axs = plt.subplots(nrows=5, ncols=2, figsize=(15, 15))  *# Plot the columns using seaborn*  sns.lineplot(x='epoch', y='train/box\_loss', data=df, ax=axs[0,0])  sns.lineplot(x='epoch', y='train/cls\_loss', data=df, ax=axs[0,1])  sns.lineplot(x='epoch', y='train/dfl\_loss', data=df, ax=axs[1,0])  sns.lineplot(x='epoch', y='metrics/precision(B)', data=df, ax=axs[1,1])  sns.lineplot(x='epoch', y='metrics/recall(B)', data=df, ax=axs[2,0])  sns.lineplot(x='epoch', y='metrics/mAP50(B)', data=df, ax=axs[2,1])  sns.lineplot(x='epoch', y='metrics/mAP50-95(B)', data=df, ax=axs[3,0])  sns.lineplot(x='epoch', y='val/box\_loss', data=df, ax=axs[3,1])  sns.lineplot(x='epoch', y='val/cls\_loss', data=df, ax=axs[4,0])  sns.lineplot(x='epoch', y='val/dfl\_loss', data=df, ax=axs[4,1])  *# Set titles and axis labels for each subplot*  axs[0,0].set(title='Train Box Loss')  axs[0,1].set(title='Train Class Loss')  axs[1,0].set(title='Train DFL Loss')  axs[1,1].set(title='Metrics Precision (B)')  axs[2,0].set(title='Metrics Recall (B)')  axs[2,1].set(title='Metrics mAP50 (B)')  axs[3,0].set(title='Metrics mAP50-95 (B)')  axs[3,1].set(title='Validation Box Loss')  axs[4,0].set(title='Validation Class Loss')  axs[4,1].set(title='Validation DFL Loss')  plt.suptitle('Training Metrics and Loss', fontsize=24)  plt.subplots\_adjust(top=0.8)  plt.tight\_layout()  plt.show() Validation of the Model By TestSet In [ ]:  *# Loading the best performing model*  Valid\_model = YOLO('/kaggle/working/runs/detect/train/weights/best.pt')  *# Evaluating the model on the testset*  metrics = Valid\_model.val(split = 'test')  In [ ]:  *# final results*  print("precision(B): ", metrics.results\_dict["metrics/precision(B)"])  print("metrics/recall(B): ", metrics.results\_dict["metrics/recall(B)"])  print("metrics/mAP50(B): ", metrics.results\_dict["metrics/mAP50(B)"])  print("metrics/mAP50-95(B): ", metrics.results\_dict["metrics/mAP50-95(B)"])  **Tip: Based on the observed results, it is evident that the accuracy of the model on both the validation and test data sets exhibits a high degree of similarity. This outcome serves as an indication that the model has been appropriately trained.** Making Predictions On Test Images In [ ]:  *# Path to the directory containing the images*  image\_dir = '/kaggle/input/cardetection/car/test/images'  *# Get a list of all image files in the directory*  image\_files = [os.path.join(image\_dir, file) for file **in** os.listdir(image\_dir) if file.endswith('.jpg')]  *# Randomly select 10 images from the directory*  random\_images = random.sample(image\_files, k=10)  for image\_path **in** random\_images:  image = cv2.imread(image\_path) *# Replace with your preferred method of reading the image*  results = Final\_model.predict([image], save=True, imgsz=416, conf=0.5, iou=0.7)  *#results.append(result)*  In [ ]:  *# View results*  for i **in** range(2,12):  plt.imshow(plt.imread(f'/kaggle/working/runs/detect/train**{**i**}**/image0.jpg'))  plt.show() Export The Final Model Of Detect Traffic Signs In [ ]:  *# Export the model*  video\_model.export(format='onnx') Try Pre-trained YOLOv8 For Detect Traffic Signs From Video In [ ]:  *# Convert mp4*  !ffmpeg -y -loglevel panic -i /kaggle/input/cardetection/output.mp4 output.mp4  *# Display the video*  Video("output.mp4", width=960) Predict By Pr-Trained YOLOv8 unfold\_moreShow hidden code  In [ ]:  *# show result*  *# Convert format*  !ffmpeg -y -loglevel panic -i /kaggle/working/runs/detect/predict/output.avi result\_out.mp4  *# Display the video*  Video("result\_out.mp4", width=960) |
|  |

# Mini Project

## Mini Project

* **Aim –** The project aims to implementation in web application for the House Price Prediction.
* **Theory –**

1. **Machine Learning Model:**

**Regression Analysis:** House price prediction is a regression problem where the aim is to predict a continuous output (price). The machine learning model uses regression techniques to analyze and infer the relationship between input features (like area, number

of bedrooms, location, etc.) and the output price.

**Model Selection:** The choice of using a machine learning framework such as sklearn is justified by their flexibility in handling various layers, and large datasets, which are typical in real estate price prediction.

1. **Flask Web Framework:**

**Flask Overview:** Flask is a micro web framework for Python, chosen for its simplicity and efficiency in building web applications. It is particularly suitable for small to medium applications and prototyping.

**Routing and View Functions:** Flask uses decorators to link functions to URLs, making it straightforward to create web pages that interact with the user. This feature is essential for taking user inputs and displaying predictions.

1. **User Interface with HTML:**

**HTML Forms:** HTML forms are used to collect user inputs, which are crucial for making predictions. The ease of integrating HTML with Python/Flask and updating it dynamically makes it an excellent choice for front-end development.

**Responsiveness and Accessibility:** The design considerations ensure that the web application is user-friendly and accessible across different devices and browsers. This inclusiveness enhances user interaction and satisfaction.

1. **Data Handling and Security:**

**Data Validation:** Before making predictions, it is critical to validate and preprocess user inputs to avoid errors and improve prediction accuracy. This involves checking for outliers, handling missing values, and ensuring data types are correct.

**Security Measures:** The application must protect sensitive data and prevent common vulnerabilities such as SQL injection and cross-site scripting (XSS). Techniques include validating user inputs, using HTTPS, and setting secure HTTP headers.

1. **Deployment Considerations:**

**Scalability:** Discuss how the Flask application can be scaled using additional tools like Gunicorn or Nginx, especially when deploying to a production environment.

**Performance Optimization:** Techniques such as caching, load balancing, and asynchronous processing are important to ensure the application handles multiple requests efficiently.

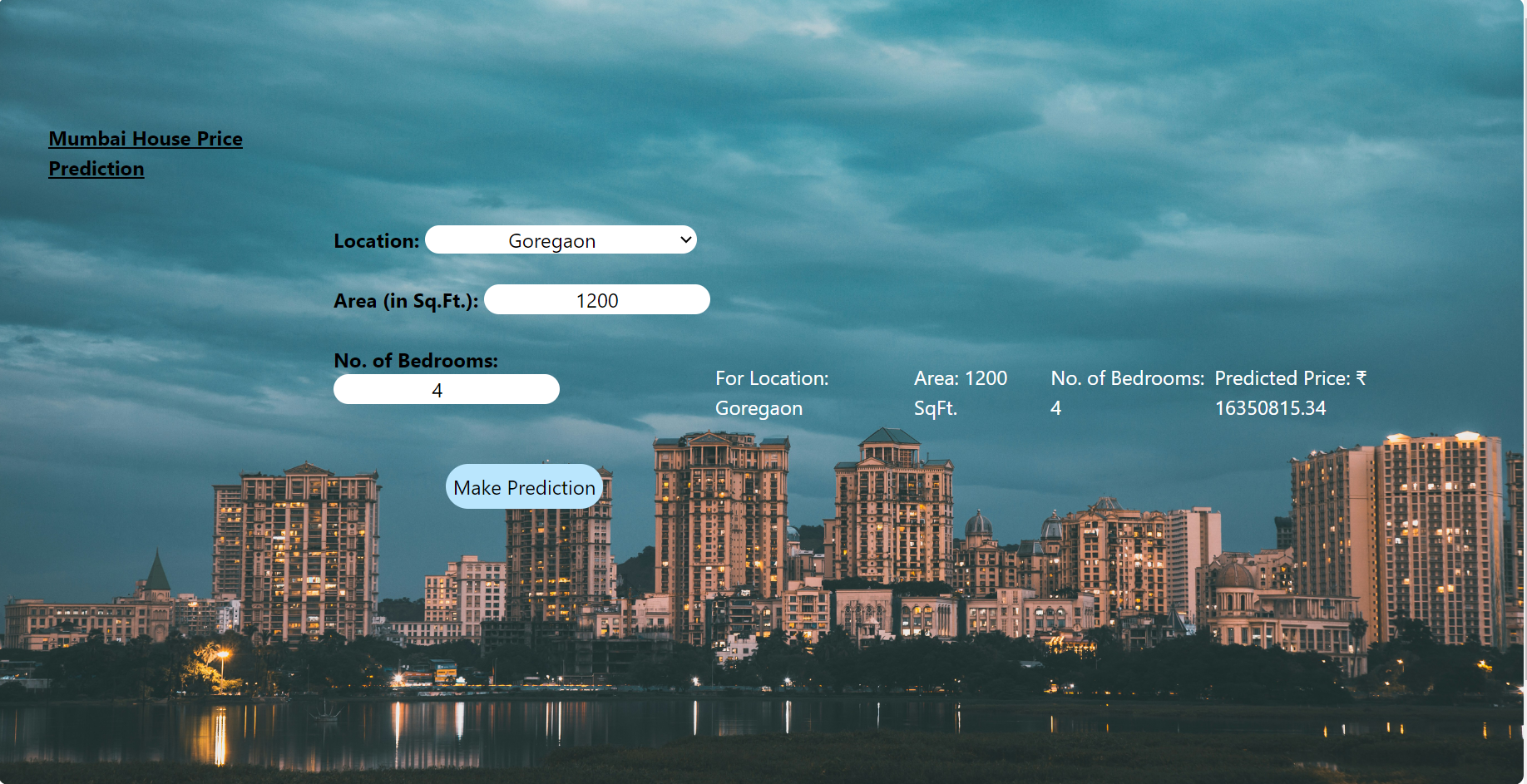
**Implementation**

**1. Setting up Flask**

**2. Backend**

**3. Frontend**

1. **Results**



**Example Use Case:**

An example use case in the theory section can describe a scenario where a user inputs several house features into the web form and receives the predicted price. The discussion can elaborate on how the machine learning model processes these inputs using the trained weights and biases to compute the output, demonstrating the practical application of theoretical concepts like neural networks and regression analysis.

**Conclusion:**

In conclusion, the theoretical foundation of the Flask web application for house price prediction hinges on understanding machine learning principles, web application architecture, user interface design, data security, and deployment strategies. This comprehensive understanding ensures that the implementation is robust, secure, and user-friendly, bridging the gap between complex data models and practical real-world applications.