Project Development Phase
Sprint - 2

Date	06 November 2022
Team ID	PNT2022TMID35960
Project Name	Project – Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

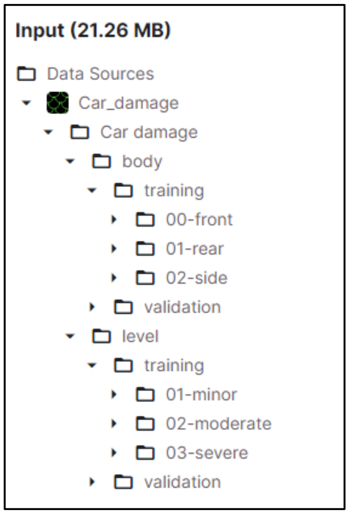
Model Building – Level of Damage:

Kaggle code Link: https://www.kaggle.com/code/balasubramaniankn/level-of-damage

Importing Phase:

```python
from tensorflow.keras.layers import Input,Dense,Flatten, Dropout
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
from matplotlib import pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
import cv2
import shutil
import random
```

Data Collection:



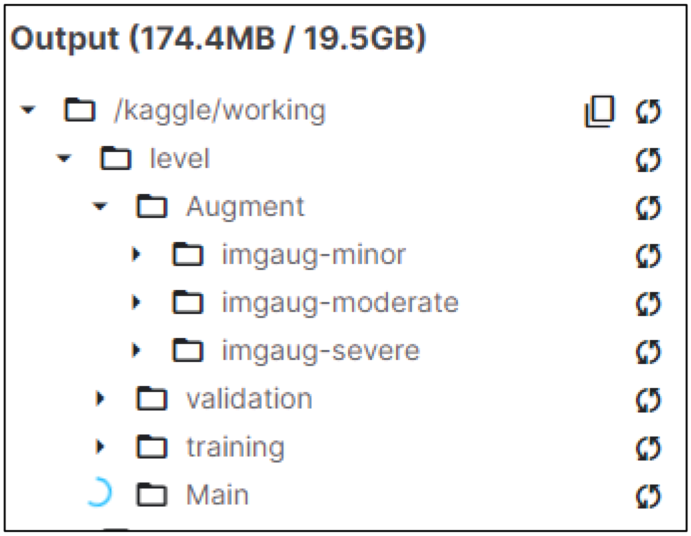
Input (21.26 MB)

- Data Sources
  - Car_damage
    - Car damage
      - body
        - training
          - 00-front
          - 01-rear
          - 02-side
        - validation
      - level
        - training
          - 01-minor
          - 02-moderate
          - 03-severe
        - validation

Creation of Directories:

Different directories are created such as Main, Augment, Training and Validation and performed move and store operations respectively.

```
# creating directory
os.mkdir('./level')
os.mkdir('./level/training')
os.mkdir('./level/training/imgaug-minor')
os.mkdir('./level/training/imgaug-moderate')
os.mkdir('./level/training/imgaug-severe')

os.mkdir('./level/Main')
os.mkdir('./level/Main/real-minor')
os.mkdir('./level/Main/real-moderate')
os.mkdir('./level/Main/real-severe')

os.mkdir('./level/validation')
os.mkdir('./level/validation/imgaug-minor')
os.mkdir('./level/validation/imgaug-moderate')
os.mkdir('./level/validation/imgaug-severe')

os.mkdir('./level/Augment')
os.mkdir('./level/Augment/imgaug-minor')
os.mkdir('./level/Augment/imgaug-moderate')
os.mkdir('./level/Augment/imgaug-severe')
```



Output (174.4MB / 19.5GB)

```
/kaggle/working
  level
    Augment
      imgaug-minor
      imgaug-moderate
      imgaug-severe
    validation
    training
    Main
```

Merging of Dataset:

To merge both training and validation image data into one in to the Main directory sub folders to perform together augmentation.

1) Training data:

```
path_minor = "../input/car-damage/Car damage/level/training/01-minor"
dir_list_mi = os.listdir(path_minor)
path_moderate = '../input/car-damage/Car damage/level/training/02-moderate'
dir_list_mo = os.listdir(path_moderate)
path_severe = '../input/car-damage/Car damage/level/training/03-severe'
dir_list_s = os.listdir(path_severe)

for j in dir_list_mi:
    IMAGE_PATH = path_minor+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Main/real-minor'
    cv2.imwrite(os.path.join(path,j), image)

for j in dir_list_mo:
    IMAGE_PATH = path_moderate+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Main/real-moderate'
    cv2.imwrite(os.path.join(path,j), image)

for j in dir_list_s:
    IMAGE_PATH = path_severe+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Main/real-severe'
    cv2.imwrite(os.path.join(path,j), image)
```

2) validation data:

```
path_minorv = "../input/car-damage/Car damage/level/validation/01-minor"
dir_list_miv = os.listdir(path_minorv)
path_moderatev = '../input/car-damage/Car damage/level/validation/02-moderate'
dir_list_mov = os.listdir(path_moderatev)
path_severev = '../input/car-damage/Car damage/level/validation/03-severe'
dir_list_sv = os.listdir(path_severev)

for j in dir_list_miv:
    IMAGE_PATH = path_minorv+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Main/real-minor'
    cv2.imwrite(os.path.join(path,j), image)

for j in dir_list_mov:
    IMAGE_PATH = path_moderatev+'/'+j
    image = cv2.imread(IMAGE_PATH)
    path = './level/Main/real-moderate'
    cv2.imwrite(os.path.join(path,j), image)

for j in dir_list_sv:
    IMAGE_PATH = path_severev+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Main/real-severe'
    cv2.imwrite(os.path.join(path,j), image)
```

Image Augmentation:

To perform shifting, Right rotation and horizontal flip on all the images and store the result in the augment directory.

```
#Augmenting and saving train level minor view images

OUTPUT_DIRECTORY = './level/Augment/imgaug-minor'

# Get the list of all files and directories
path_minor = "./level/Main/real-minor"
dir_list = os.listdir(path_minor)
for j in dir_list:
    IMAGE_PATH = path_minor+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Augment/imgaug-minor'
    cv2.imwrite(os.path.join(path,j), image)

    image = load_img(IMAGE_PATH)
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)

    datagen_shift = ImageDataGenerator(height_shift_range=0.2, width_shift_range=
0.2)
    PREFIX = 'Shifted'
    imGen = datagen_shift.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTOR
Y,
                        save_prefix=PREFIX, save_format='jpg')
    for i in range(6):
        batch = imGen.next()

    datagen_rot = ImageDataGenerator(rotation_range=30)
    PREFIX = 'Rotated'
    imGen = datagen_rot.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                        save_prefix=PREFIX, save_format='jpg')
    for i in range(6):
        batch = imGen.next()

    datagen_hf = ImageDataGenerator(horizontal_flip=True)
    PREFIX = 'Hortizonal_flip'
    imGen = datagen_hf.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                            save_prefix=PREFIX, save_format='jpg')
    for i in range(1):
        batch = imGen.next()
```

```
#Augmenting and saving train level moderate view images

OUTPUT_DIRECTORY = './level/Augment/imgaug-moderate'

# Get the list of all files and directories
path_mod = "./level/Main/real-moderate"
dir_list = os.listdir(path_mod)
for j in dir_list:
    IMAGE_PATH = path_mod+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Augment/imgaug-moderate'
    cv2.imwrite(os.path.join(path,j), image)

    image = load_img(IMAGE_PATH)
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)

    datagen_shift = ImageDataGenerator(height_shift_range=0.2, width_shift_range=
0.2)
    PREFIX = 'Shifted'
    imGen = datagen_shift.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTOR
Y,
                            save_prefix=PREFIX, save_format='jpg')
    for i in range(6):
        batch = imGen.next()

    datagen_rot = ImageDataGenerator(rotation_range=30)
    PREFIX = 'Rotated'
    imGen = datagen_rot.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                            save_prefix=PREFIX, save_format='jpg')
    for i in range(6):
        batch = imGen.next()

    datagen_hf = ImageDataGenerator(horizontal_flip=True)
    PREFIX = 'Hortizonal_flip'
    imGen = datagen_hf.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                            save_prefix=PREFIX, save_format='jpg')
    for i in range(1):
        batch = imGen.next()
```

```
#Augmenting and saving train level severe view images

OUTPUT_DIRECTORY = './level/Augment/imgaug-severe'
```

```
# Get the list of all files and directories
path_sev = "./level/Main/real-severe"
dir_list = os.listdir(path_sev)
for j in dir_list:
    IMAGE_PATH = path_sev+'/'+j

    image = cv2.imread(IMAGE_PATH)
    path = './level/Augment/imgaug-side'
    cv2.imwrite(os.path.join(path,j), image)

    image = load_img(IMAGE_PATH)
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)

    datagen_shift = ImageDataGenerator(height_shift_range=0.2, width_shift_range=0.2)
    PREFIX = 'Shifted'
    imGen = datagen_shift.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                        save_prefix=PREFIX, save_format='jpg')
    for i in range(6):
        batch = imGen.next()

    datagen_rot = ImageDataGenerator(rotation_range=30)
    PREFIX = 'Rotated'
    imGen = datagen_rot.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                        save_prefix=PREFIX, save_format='jpg')
    for i in range(6):
        batch = imGen.next()

    datagen_hf = ImageDataGenerator(horizontal_flip=True)
    PREFIX = 'Hortizonal_flip'
    imGen = datagen_hf.flow(image, batch_size=1, save_to_dir = OUTPUT_DIRECTORY,
                        save_prefix=PREFIX, save_format='jpg')
    for i in range(1):
        batch = imGen.next()
```

Splitting of Dataset:

To split the augmented image in the ratio of 80:20 and store it in respective folders and sub folders.

```
# Split the minor level data in 80:20 ratio
no_of_minor = os.listdir('./level/Augment/imgaug-minor')
len(no_of_minor)
augment_data =  './level/Augment/imgaug-minor'
for f in no_of_minor:
    if random.random() > 0.80:
        shutil.move(f'{augment_data}/{f}','./level/validation/imgaug-minor' )
    else:
        shutil.move(f'{augment_data}/{f}','./level/training/imgaug-minor')
```

```
# Split the moderate level data in 80:20 ratio
no_of_mod = os.listdir('./level/Augment/imgaug-moderate')

augment_data =  './level/Augment/imgaug-moderate'
for f in no_of_mod:
    if random.random() > 0.80:
        shutil.move(f'{augment_data}/{f}','./level/validation/imgaug-moderate' )
    else:
        shutil.move(f'{augment_data}/{f}','./level/training/imgaug-moderate')
```

```
# Split the severe level data in 80:20 ratio
no_of_sev = os.listdir('./level/Augment/imgaug-severe')

augment_data =  './level/Augment/imgaug-severe'
for f in no_of_sev:
    if random.random() > 0.80:
        shutil.move(f'{augment_data}/{f}','./level/validation/imgaug-severe' )
    else:
        shutil.move(f'{augment_data}/{f}','./level/training/imgaug-severe')
```

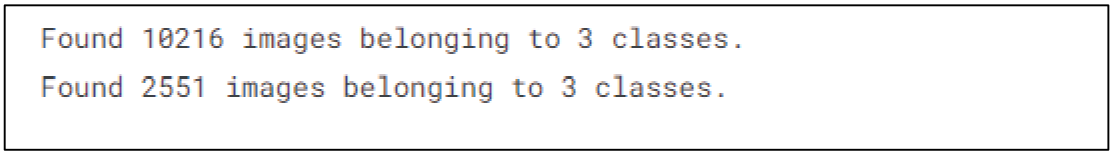
Flow from Directory – Augmentation:

To store the path of the train and validating data.

```
img_size = [224,224] #List which stores the resolution
main_train = './level/training' #Stores the path of the train directory
main_test = './level/validation' #Stores the path of the test directory
```

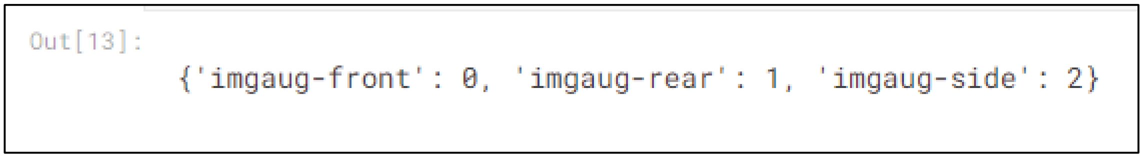
To modify the train and validation data with respect to the properties.

```
train_datagen = ImageDataGenerator(rescale = 1/255.0)

test_datagen = ImageDataGenerator(rescale = 1/255.0)

# flow_from_directory() is used to convert all the images in the specific directo
ry
training_set = train_datagen.flow_from_directory(directory = main_train,
                                                 target_size = (224,224),
                                                 batch_size = 100,
                                                 )

test_set = test_datagen.flow_from_directory(directory = main_test,
                                                 target_size = (224,224),
                                                 batch_size = 100,
                                                 )
```



```
Found 10216 images belonging to 3 classes.
Found 2551 images belonging to 3 classes.
```

```
# Class_indices will display the respective class value
training_set.class_indices
```

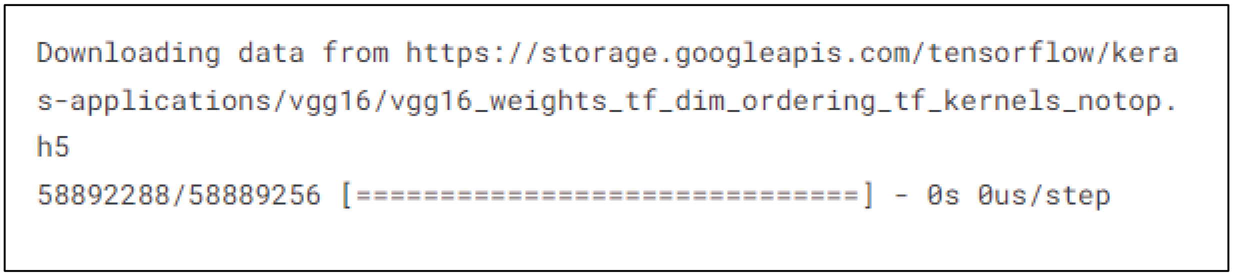


```
Out[13]:
        {'imgaug-front': 0, 'imgaug-rear': 1, 'imgaug-side': 2}
```

Model Building:

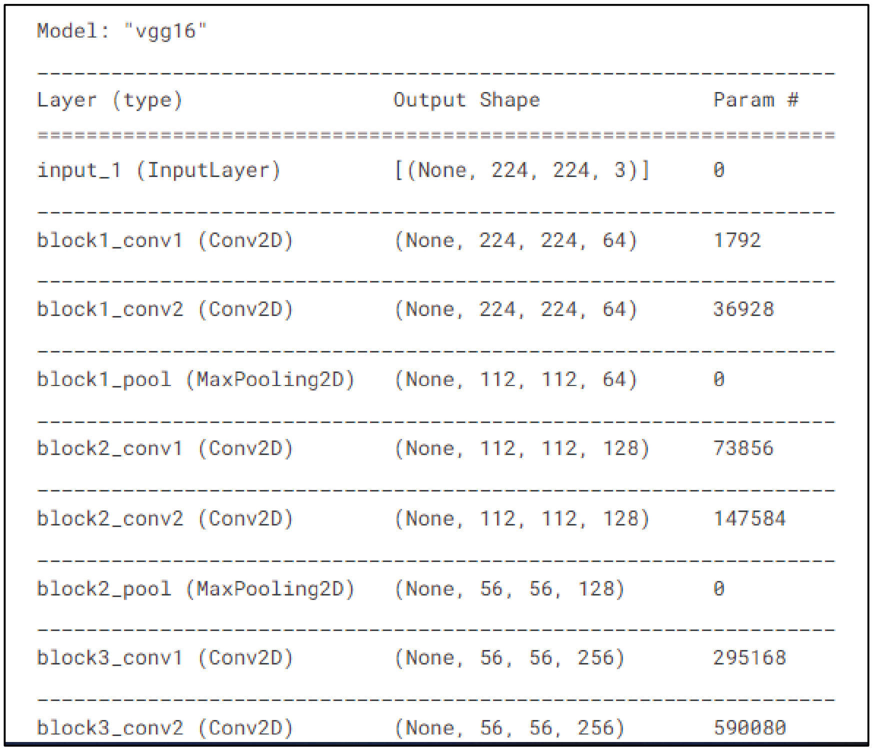
- Loading the VGG16 pre trained model.
- Include_top - this specifies whether the final layer before the output layer has to be include.
- If included then there will be 1000 number of classes at the output. # Weights are trained using imagenet

```
vgg_model = VGG16(include_top=False,
    weights="imagenet",
    input_shape=img_size + [3])
```

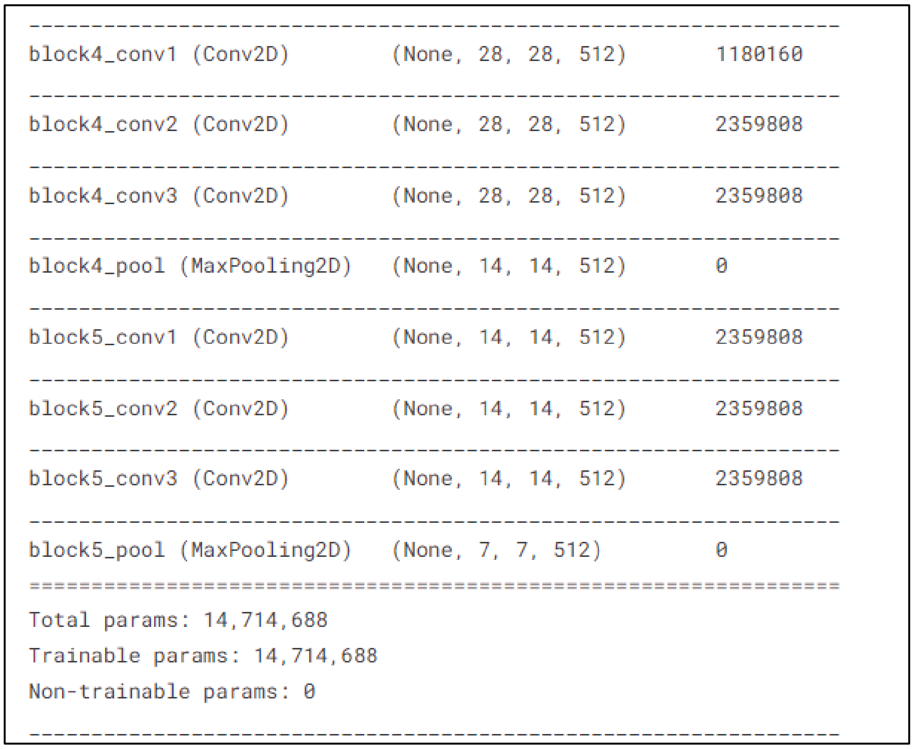


```
Downloading data from https://storage.googleapis.com/tensorflow/kera
s-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
h5
58892288/58889256 [==============================] - 0s 0us/step
```

```
# To print the hidden layer summary of vgg model without top layer
vgg_model.summary()
```



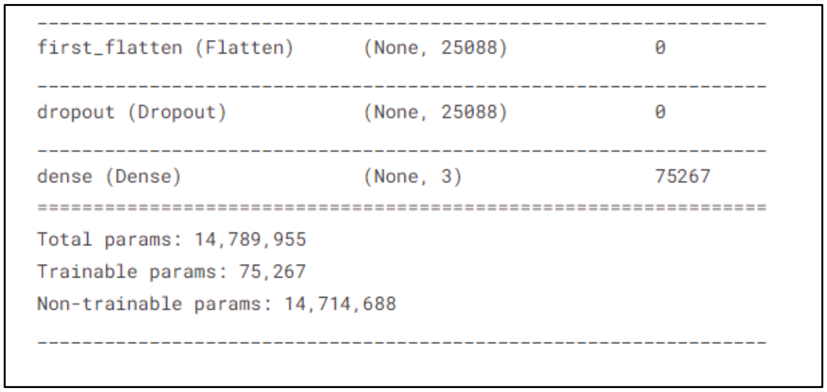
```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
```



```
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

```
# To fix the weights of the pre trained model
for lay in vgg_model.layers:
    lay.trainable = False
```

```
# Flatten() is used to convert the last layer to vector or as fully connected
x = Flatten(name="first_flatten")(vgg_model.output)
# Dense() layer is added such that it outputs only two classess
# Softmax activation layer produces probabilities for different classess.
x = Dropout(0.5)(x)
pred = Dense(3,activation='softmax')(x)
# Model() is used to group layers
model = Model(inputs=vgg_model.input,outputs=pred)
model.summary()
```



```
_________________________________________________________________
first_flatten (Flatten)      (None, 25088)             0

_________________________________________________________________
dropout (Dropout)            (None, 25088)             0

_________________________________________________________________
dense (Dense)                (None, 3)                 75267
=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688

_________________________________________________________________
```

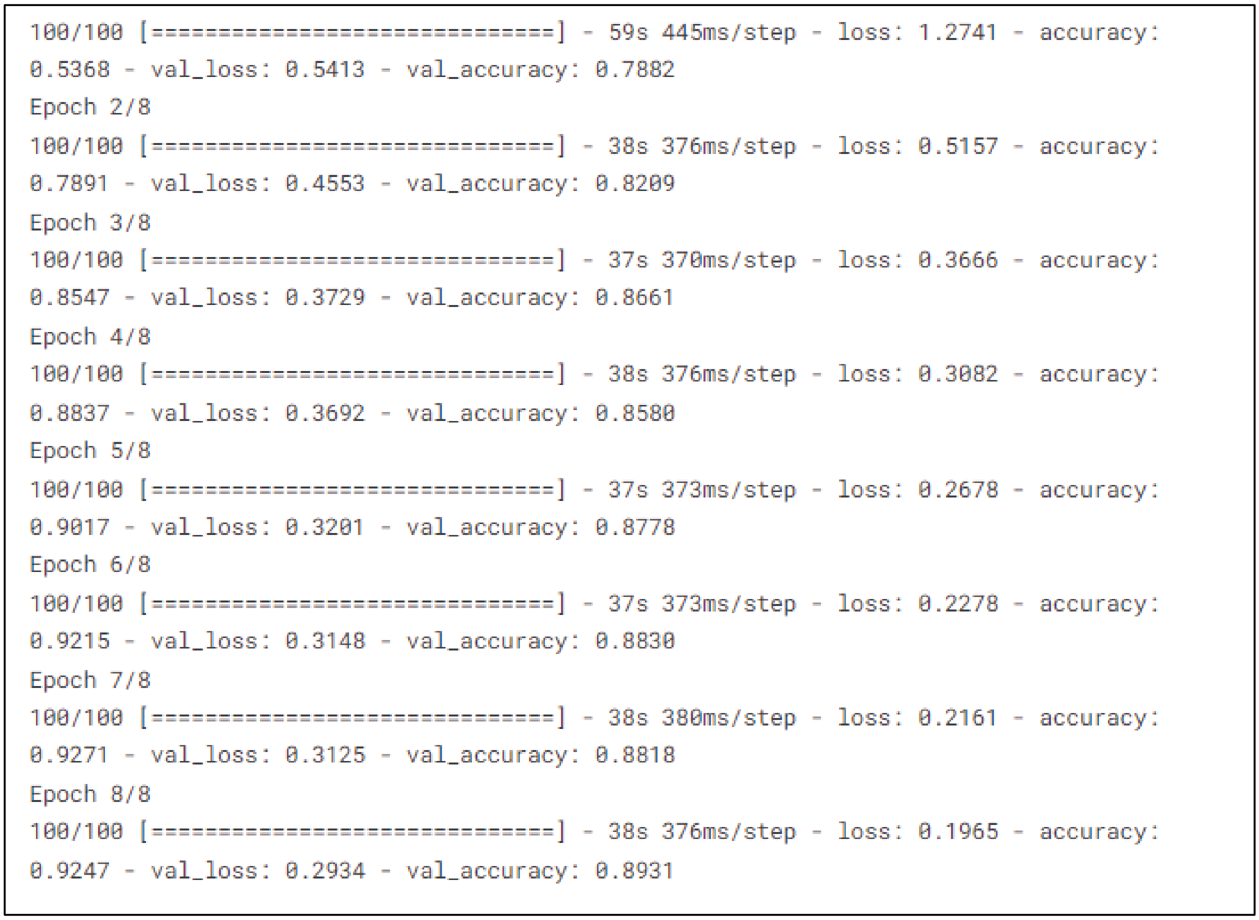
Model Fitting:

- Loss function is used to find the errors or deviations in learning process.
- Optimizer is used to optimize the input weights.
- Metrics is used to measure the performance

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

Training the model for 8 epochs:

```
#fit() is used to train the model
mod = model.fit( training_set,
                validation_data=test_set,
                epochs=8,
                steps_per_epoch=len(training_set),
                validation_steps=len(test_set)
               )
```



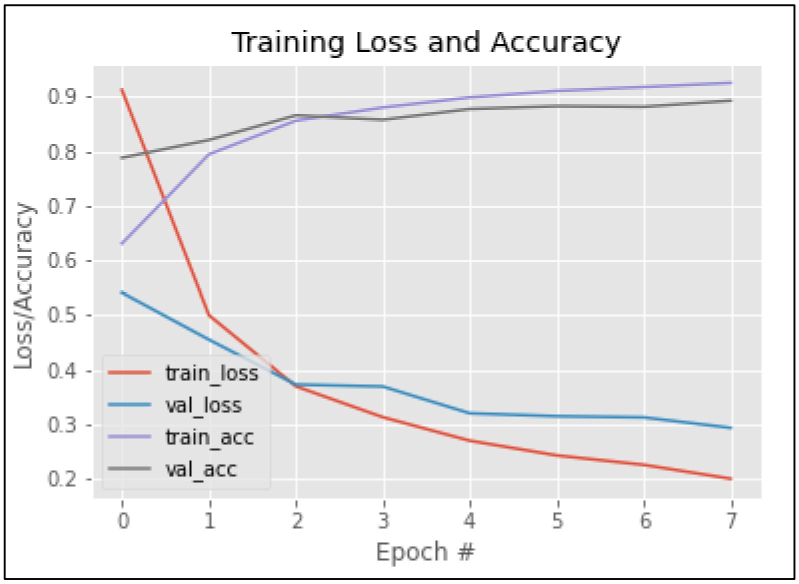
```
100/100 [==============================] - 59s 445ms/step - loss: 1.2741 - accuracy:
0.5368 - val_loss: 0.5413 - val_accuracy: 0.7882
Epoch 2/8
100/100 [==============================] - 38s 376ms/step - loss: 0.5157 - accuracy:
0.7891 - val_loss: 0.4553 - val_accuracy: 0.8209
Epoch 3/8
100/100 [==============================] - 37s 370ms/step - loss: 0.3666 - accuracy:
0.8547 - val_loss: 0.3729 - val_accuracy: 0.8661
Epoch 4/8
100/100 [==============================] - 38s 376ms/step - loss: 0.3082 - accuracy:
0.8837 - val_loss: 0.3692 - val_accuracy: 0.8580
Epoch 5/8
100/100 [==============================] - 37s 373ms/step - loss: 0.2678 - accuracy:
0.9017 - val_loss: 0.3201 - val_accuracy: 0.8778
Epoch 6/8
100/100 [==============================] - 37s 373ms/step - loss: 0.2278 - accuracy:
0.9215 - val_loss: 0.3148 - val_accuracy: 0.8830
Epoch 7/8
100/100 [==============================] - 38s 380ms/step - loss: 0.2161 - accuracy:
0.9271 - val_loss: 0.3125 - val_accuracy: 0.8818
Epoch 8/8
100/100 [==============================] - 38s 376ms/step - loss: 0.1965 - accuracy:
0.9247 - val_loss: 0.2934 - val_accuracy: 0.8931
```

Saving the Model:

```
# To save the particular model in .h5 format
import tensorflow as tf
from tensorflow.keras.models import load_model
model.save('vggmodelfinallevel.h5')
```

Model Visualization:

```
from matplotlib import pyplot as plt
N = 8
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), mod.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), mod.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), mod.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), mod.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('grp.png')
```

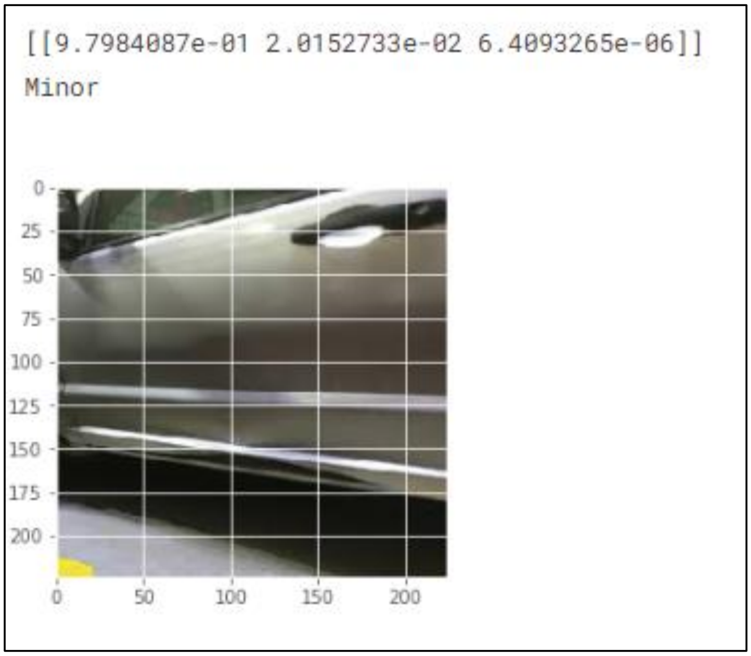


It is evident from the graph that the model is not overfitting and it near to best fit. The validation accuracy obtained is 89%.

Model Testing:

1)

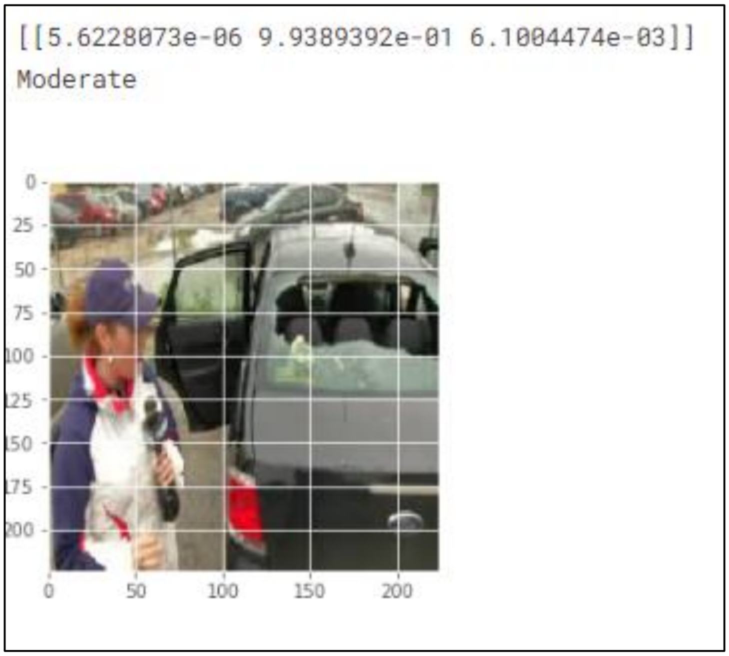
```
from tensorflow.keras.preprocessing import image
img12 =image.load_img('../input/car-damage/Car damage/level/validation/01-minor/0010.JPEG',target_size=(224,224))
plt.imshow(img12)
img12 = image.img_to_array(img12)
img12 = img12/255.0
img12 = np.expand_dims(img12,axis=0)
pred1 = model.predict(img12)
print(pred1)
pred1 = np.argmax(pred1,axis=1)



if pred1[0] == 1:
    print("Moderate")
elif pred1[0] == 0:
    print("Minor")
else:
    print("Severe")
```



```
[[9.7984087e-01 2.0152733e-02 6.4093265e-06]]
Minor
```

2)

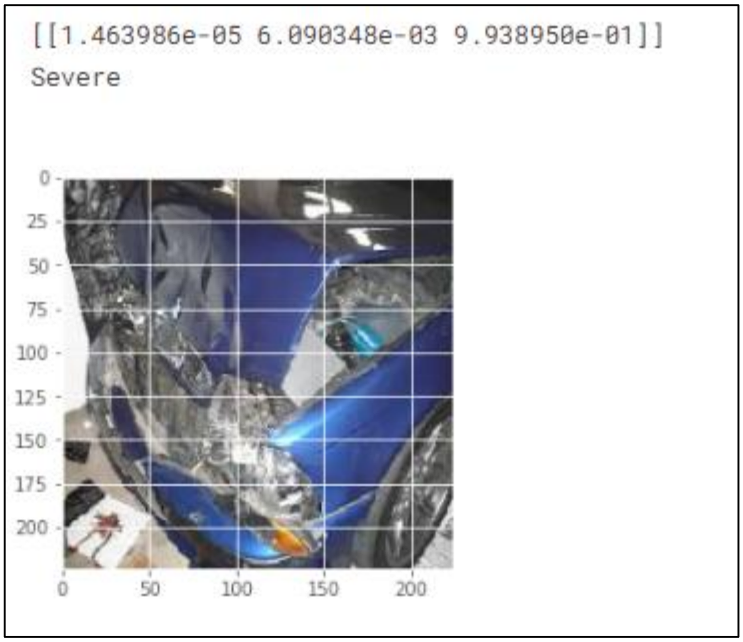
```
img12 =image.load_img('../input/car-damage/Car damage/level/validation/02-moderat
e/0006.JPEG',target_size=(224,224))
plt.imshow(img12)
img12 = image.img_to_array(img12)
img12 = img12/255.0
img12 = np.expand_dims(img12,axis=0)
pred1 = model.predict(img12)
print(pred1)
pred1 = np.argmax(pred1,axis=1)



if pred1[0] == 1:
    print("Moderate")
elif pred1[0] == 0:
    print("Minor")
else:
    print("Severe")
```



```
[[5.6228073e-06 9.9389392e-01 6.1004474e-03]]
Moderate
```

3)

```
img12 =image.load_img('../input/car-damage/Car damage/level/validation/03-severe/
0009.JPEG',target_size=(224,224))
plt.imshow(img12)
img12 = image.img_to_array(img12)
img12 = img12/255.0
img12 = np.expand_dims(img12,axis=0)
pred1 = model.predict(img12)
print(pred1)
pred1 = np.argmax(pred1,axis=1)


if pred1[0] == 1:
    print("Moderate")
elif pred1[0] == 0:
    print("Minor")
else:
    print("Severe")
```

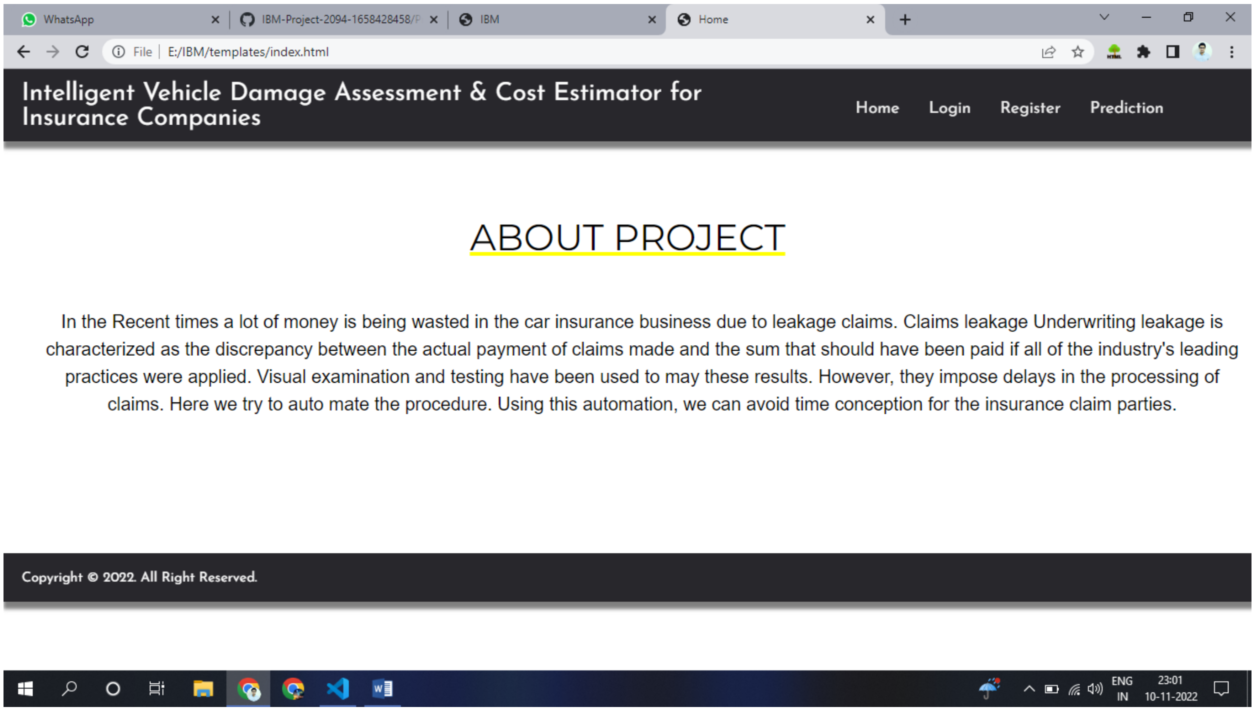


```
[[1.463986e-05 6.090348e-03 9.938950e-01]]
Severe
```

Building Application:

Creation of web pages without integrating with the flask module:

Index.html



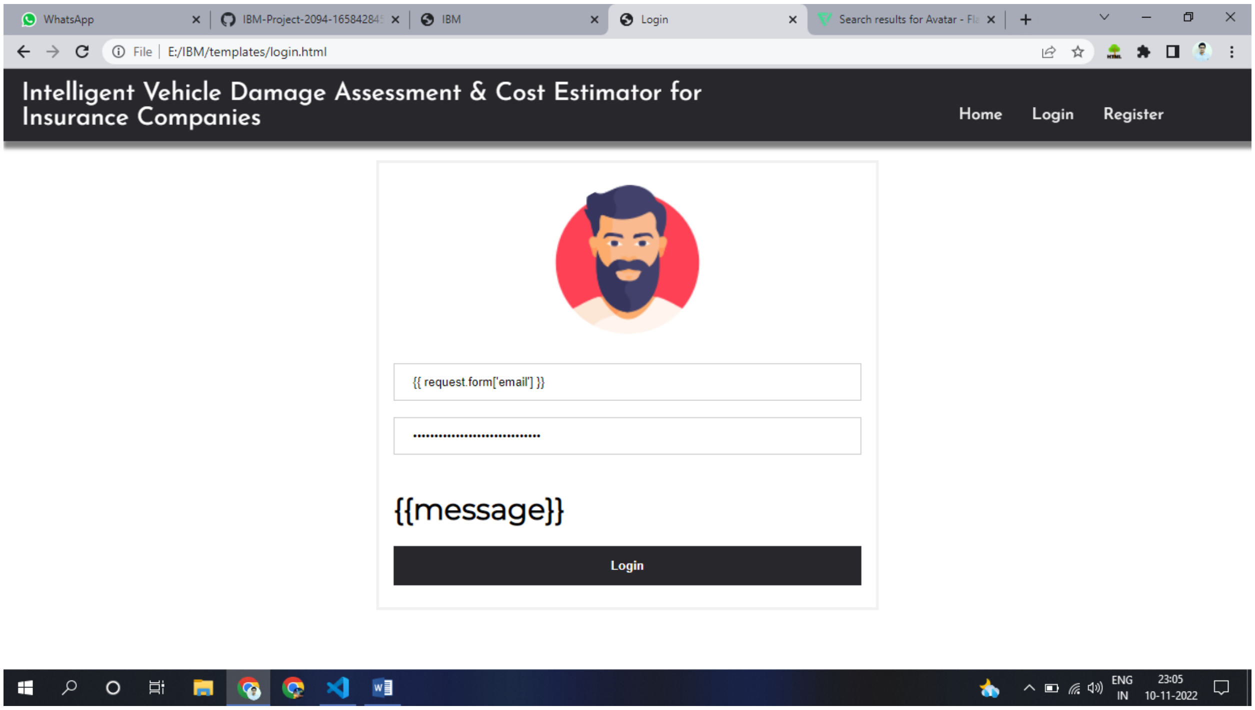
Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

ABOUT PROJECT

In the Recent times a lot of money is being wasted in the car insurance business due to leakage claims. Claims leakage Underwriting leakage is characterized as the discrepancy between the actual payment of claims made and the sum that should have been paid if all of the industry's leading practices were applied. Visual examination and testing have been used to may these results. However, they impose delays in the processing of claims. Here we try to auto mate the procedure. Using this automation, we can avoid time conception for the insurance claim parties.

Copyright © 2022. All Right Reserved.

Login.html


WhatsApp
IBM-Project-2094-16584284
IBM
Login
Search results for Avatar - Fl
File | E:/IBM/templates/login.html
Home
Login
Register
Login
ENG IN
23:05
10-11-2022

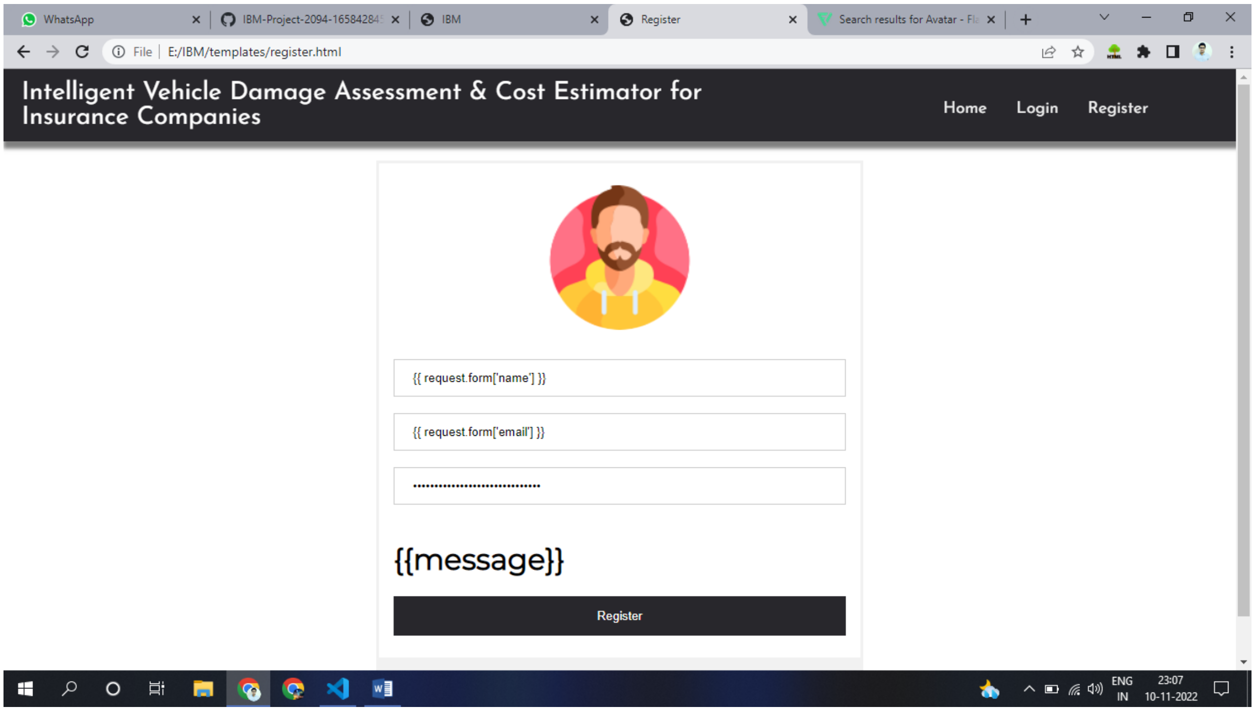
Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

{{ request.form['email'] }}

································

{{message}}

Register.html


WhatsApp
IBM-Project-2094-16584284
IBM
Register
Search results for Avatar - Fl
File | E:/IBM/templates/register.html
Home
Login
Register
Register
ENG IN
23:07
10-11-2022

Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

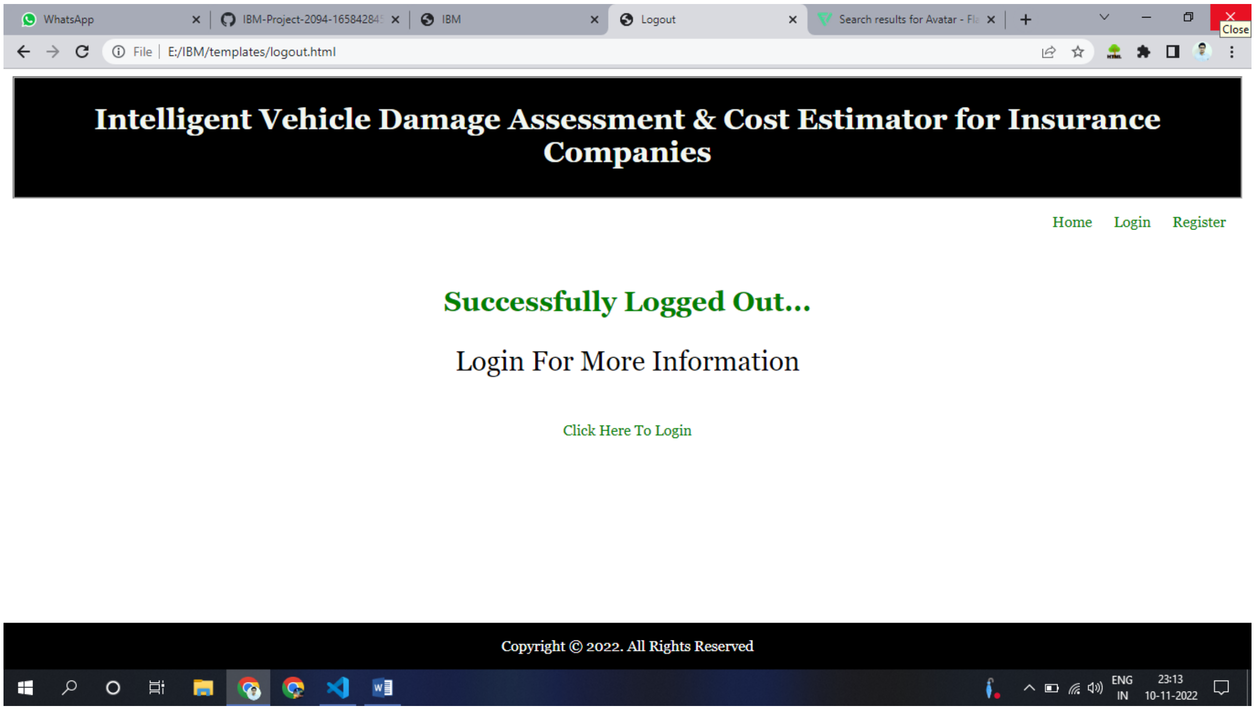
{{ request.form['name'] }}

{{ request.form['email'] }}

································

{{message}}

Logout.html


WhatsApp
IBM-Project-2094-16584284
IBM
Logout
Search results for Avatar - Fl
Close
File
E:/IBM/templates/logout.html
ENG
IN
23:13
10-11-2022

Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

Home Login Register

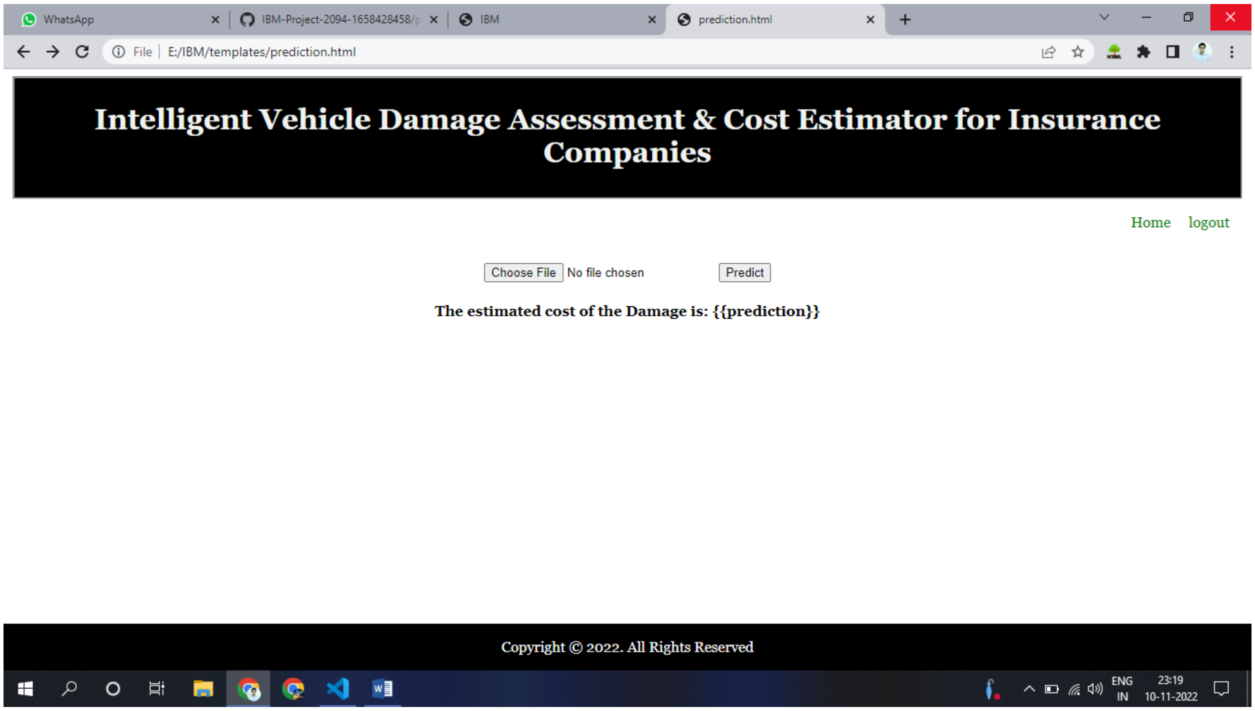
Successfully Logged Out...

Login For More Information

Click Here To Login

Copyright © 2022. All Rights Reserved

Prediction.html


WhatsApp
IBM-Project-2094-1658428458/
IBM
prediction.html
File
E:/IBM/templates/prediction.html
ENG
IN
23:19
10-11-2022

Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

Home logout

Choose File No file chosen Predict

The estimated cost of the Damage is: {{prediction}}

Copyright © 2022. All Rights Reserved