INTELLIGENT VEHICLE DAMAGE ASSESSMENT AND COST ESTIMATOR FOR INSURANCE COMPANIES

ABSTRACT

The motor insurance sector loses a lot of money as a result of leakage claims. The gap between the amount actually paid for claims and the amount that would have been paid had all of the best practices in the industry been followed is known as underwriting leakage. These results have been reached using both testing and visual assessment. However, they do delay the processing of claims. By reducing loss adjustment costs, improvements in the First Notice of Loss and the speed with which claims are examined and evaluated might save a lot of money in the automobile insurance claims process. Car damage is automatically identified and classified using advanced picture analysis and pattern recognition technology, a method for automatically locating the damaged area by comparing photos of the automobile from before and after an accident. This project's proposed a CNN model that can recognise a car's damage area. If users upload images, the model can evaluate damage (be it a dent or scratch from an object), and it can also estimate the extent of damage. Insurance firms can handle claims more efficiently as a result. When accepting a car loan, particularly one for a used vehicle, lenders may also consider this model.

1. INTRODUCTION

PROJECT OVERVIEW

Vehicles are significantly rising in today's globe. Because there are more cars on the road, accidents happen more frequently because individuals are driving them at high speeds. When an accident occurs, the people file a claim with their auto insurance for the necessary funds to repair the car, because to inaccurate claims, the

corporation behaves improperly and doesn’t make payments now. This occurs as a result of claims leakage, which is the discrepancy between the sums secured by the firm and the sums that it should have secured in accordance with the claims. Even if the car's damage is easily seen, the claim procedure will take longer than usual in accordance with company policy. Despite the company's best efforts, there is a delay in the claims procedure. Differentiate the suggested approach to perhaps speed up the process of assessing automotive damage. Instead of taking hours to accomplish automotive damage detection if it were visually inspected, a system may perform it in a minute by just providing a picture of a damaged vehicle. The system can determine the analysis of the damage, the position of the damage, and the degree of the damage using machine learning and computer vision.

PURPOSE

Today's world is seeing a substantial increase in automobiles. Because there are more automobiles on the road and more people are driving them at high speeds, accidents happen more frequently. When an accident happens, the parties involved submit a claim with their auto insurance to obtain the money needed to repair the vehicle since, according to false claims, the company acts inappropriately and withholds payments.

2. LITERATURE REVIEW

EXISTING PROBLEM

TITLE: Convolutional Neural Networks for vehicle damage detection,2021
AUTHOR NAME: R.E. Ruitenbeek

Vehicle damage is becoming an increasing liability for shared mobility providers. The high number of driver handovers necessitates the use of an accurate and quick inspection system capable of detecting minor damage and categorising it. To address this, a damage detection model is created that locates vehicle damages and categorizes them into twelve groups. To improve detection performance, multiple deep learning algorithms are used, and the effect of various transfer learning and training strategies is evaluated. The final model, which was trained on over 10,000 damage photos, can detect minor defects in a variety of environments, including water and dirt. A performance evaluation using domain experts reveals that the model performs comparably. Furthermore, the model is tested in a specially designed light street, demonstrating how strong reflections complicate detection performance.

2.1.3. TITLE:
Deep Learning Based Car Damage Detection, Classification and Severity
AUTHOR NAME: Ritik Gandhi1, 2021

Because it is a manual procedure, resolving a claim in the accident insurance sector takes time, and there is a gap between the ideal and real settlement. We are using deep learning models to not only speed up the process, but also to deliver better customer service and boost insurance company profitability. In this paper, we use multiple pre trained models such as VGG 16, VGG 19, Resnet50, and DENSENET to choose the top performing models. We first use the Resnet50 model to determine whether or not the automobile is damaged, and if it is, we utilise the WPOD-net model to identify the licence plate. The YOLO model is used to detect the affected region. Finally, the damage severity is implemented using the DENSENET model. We discovered that transfer learning outperforms fine-tuning after applying multiple models. Furthermore, we present a framework that incorporates all of this into a single application, assisting in the automation of the insurance sector.

Car Damage Assessment to Automate Insurance Claim, 2022 AUTHOR NAME: Siddhant Gole

Car damage inspection is an essential stage in claim sanctioning, and the procedure is frequently delayed and erroneous, resulting in claim leakages. Our task is to create a web application connected with a deep learning model that receives user input in the form of photographs of damaged automobiles and assesses the damage to provide a cost report that the firm can use to approve the first reimbursement. To detect and localise the damaged regions, the model employs the MASK R-CNN algorithm in conjunction with Faster RCNN. The device also includes a security module that detects and stores the vehicle's licence plate, body type, and logo data for verification. Our goal is to develop a system that can detect damaged parts of a car using images and generate a cost analysis report that the company can use to sanction the insurance amount. The task would be to develop an end-to-end system for detecting and classifying types of damage via images, as well as to implement a car number plate, body type, and logo detection system to verify car details.

Using Machine Learning Models To Compare Various Resampling Methods In Predicting Insurance Fraud, 2021 AUTHOR NAME: Ruixing Ming

Insurance fraud is one of the most prevalent kinds of fraud. In particular, the cost of automotive insurance fraud is significant for property insurance companies and has a longterm influence on insurance businesses' pricing tactics. And Car insurance fraud detection has become required in order to reduce insurance prices. Although predictive models for detecting insurance fraud are widely used in practise, there are few published research on the use of machine learning algorithms to identify insurance fraud, most likely due to a lack of available data. Evaluate 13 machine learning approaches in this paper using real-world data. Predicting insurance fraud has become a big difficulty due to the uneven datasets in this domain. Because our

data consists primarily of "non-fraud claims" with a minor number of "fraud claims." As a result, classification models predict fraud poorly; thus, the current study seeks to propose an approach that improves machine learning algorithms' results by using resampling techniques, such as Random over Sampler, Random under Sampler, and hybrid methods, to address the issue of unbalanced data.

Evaluation of deep learning algorithms for semantic segmentation of car parts, 2021 AUTHOR NAME: Kitsuchart Pasupa

One of the most significant operations in the auto insurance industry is the evaluation of accident-damaged vehicles. Currently, each fundamental component must be manually examined. It is believed that in the future, a smart device will be able to do this evaluation more effectively. We analysed and compared five deep learning algorithms for semantic segmentation of automobile parts in this work. Mask R-CNN served as the baseline reference method, while the other algorithms were HTC, CBNet, PANet, and GCNet. These five algorithms were used to do instance segmentation runs. HTC's ResNet-50 algorithm was the best for segmentation on various types of cars such as sedans, trucks, and SUVs. On our initial data set, it attained a mean average accuracy of 55.2 when distinct labels were allocated to the left and right sides, and 59.1 when a single label was assigned to both sides. Furthermore, the models from each method were verified for robustness by running them on photos of components in a real-world setting with varying weather conditions such as snow, frost, fog, and different lighting situations. When left and right sides were assigned different labels, GCNet achieved a mean performance under corruption, mPC = 35.2, and a relative degradation of performance on corrupted data, compared to clean data (rPC), of 64.4%, and mPC = 38.1 and rPC = 69.6% when left and right sides were considered the same part.

REFERENCES

[1]. R.E. Ruitenbeek, Convolutional Neural Networks for vehicle damage detection, 2021

[2]. Ritik Gandhi1Deep Learning Based Car Damage Detection, Classification and Severity, 2021

[3]. Siddhant Gole, Car Damage Assessment to Automate Insurance Claim, 2022

[4]. Ruixing Ming, Using Machine Learning Models To Compare Various Resampling Methods In Predicting Insurance Fraud, 2021

[5]. Kitsuchart Pasupa, Evaluation of deep learning algorithms for semantic segmentation of car parts, 2021

PROBLEM STATEMENT DEFINITION

In existing system, the procedure of making an insurance claim for an automobile is laborious, and there is a delay before the first reimbursement is authorised. Insurance firms lose millions of dollars each year due to claim leakage as a result of the expansion of the vehicle sector and the daily rise in the number of accidents. The discrepancy between the company's actual spending and what they should have really spent is known as claim leakage. Ineffective claim processing, erroneous payments, human error such as a lack of quality control or poor customer service or even claim fraud may be to blame for this. Auditing closed claim files is the only way to find claim leakage.

3. IDEATION & PROPOSED SOLUTION

EMPATHY MAP CANVAS

IDEATION & BRAINSTORMING

PROPOSED SOLUTION

The proposed approach collects photographs of a person's damaged automobile, then utilises those images as input for a deep learning model that use image processing to recognise the elements of the image and determine the percentage of the vehicles'' damage. After then, the images are separated into two groups: replace and repair. When the damage percentage is less than 80, the damaged part must be replaced; however, in the other case, the compensation amount is set depending on the damage percentage. Finally, it generates a comprehensive analysis report on the vehicle that is used to ask the insurance company for payment.

PROBLEM SOLUTION FIT

There is no systematic approach to receive a rapid answer from an insurance company. A week of waiting is required. The proposed solution should enable consumers to contact with the insurance provider and receive payments both online and offline. After uploading the damaged image and determining the extent of the damage, the user may obtain insurance only if the

4. ANALYSIS REQUIREMENT

FUNCTIONAL REQUIREMENT **Framework Creation:**

This approach provides a way for evaluating vehicle damage that insurance companies may utilise when processing claims. This module offered a framework for submitting a vehicle's damaged parts and requesting insurance from an organisation. The dataset needed to train the Damage Detection and it has prepared by an admin. In order to make the images useful for training, they were manually

annotated; damages were categorised into 7 distinct types such as Door Dent,

Bumper Dent, Body Scratch, Broken Windshield, Broken Glass, Broken Lights and Smash By modifying its settings and loading the learned dataset, the model was set up to train on user data.

Object Detection

Employ a specially trained CNN model utilising transfer learning on to identify the object. This model takes different forms of damage into account validation sets such as Bumper Dent, Bumper Scratch, Door Dent, Door Scratch, Glass Shattered, Head Lamp, Tail Lamp, Undamaged, etc. The classification of car damage severity is as follows: Minor Damage which typically involves slight damage to the vehicle that does not impede the vehicle to cause severe injuries. It includes the headlight scratches, dents and digs in the hood or windshield, from gravel or debris, scratches in the paint. Moderate Damage which deals with any kind of damage that impairs the functionality of the vehicle in any way is moderate damage. It involves large dents in hood, fender or door of a car. Even if the airbags are deployed during collision, then it comes under moderate damage. Severe Damage – Structural damages such as bent or twisted frames, broken/bent axels, and missing pieces of the vehicles and in some cases even the destruction of airbags. These types of damages are a big threat to the human life.

Damage Detection:

To locate damaged areas in a picture and create a bounding box around each object found, object localization is used which combines object localisation and classification to provide a bounding box and a class for each item for object detection. Use CNN to generate a convolutional features map from an image to forecast the class and bounding box of an item.

If the car is undamaged then it simply detects it and if it's a damaged one, then there are further localizations made models. The model shows accuracy on the validation

set. To automate such a system, the easiest method would be to build a Convolution Neural Network model capable of accepting images from the user and determining the location and severity of the damage. The model is required to pass through multiple checks would first ensure that given image is that of a car and then to ensure that it is in fact damaged. These are the gate checks before the analysis begins. Once all the gate checks have been validated, the damage check will commence. The model will predict the location of the damage as in front, side or rear, and the severity of such damage as in minor, moderate or severe.

Claim Insurance

The procedure of claiming insurance is done by persons who are in need. For access to the company's insurance, the user must register and authenticate. After that, users may access their insurance information and submit an insurance claim request. The request for an insurance claim can be viewed and approved by the insurance company. Once the damaged image has been uploaded and the degree of the damage has been determined, the user may receive insurance only if the firm accepts the damaged image and the condition is greater than 80%

NON FUNCTIONAL REQUIREMENTS

Usability

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Security

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

Reliability

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.

5. PROJECT DESIGN

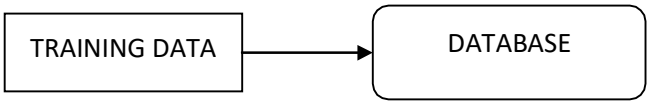
DATA FLOW DIAGRAMS

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections

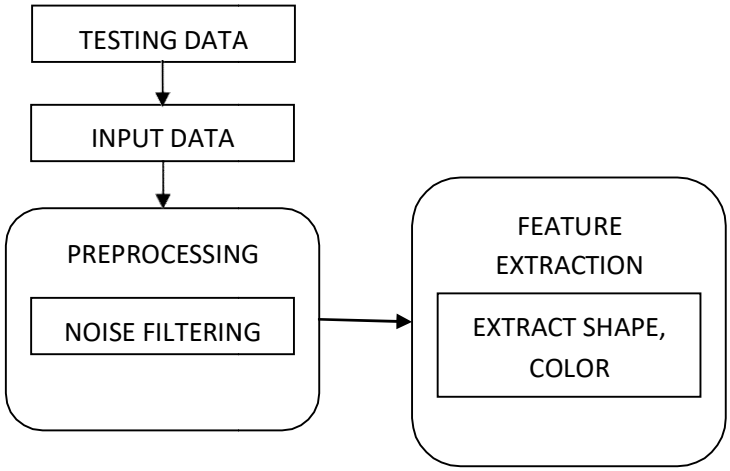
relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

LEVEL 11

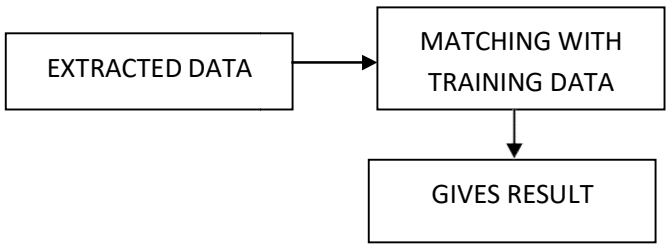
The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.



A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each domain boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows 'what' input data enters the domain, 'what' logical processes the domain applies to that data, and 'what' output data leaves the domain. Essentially, a DFD is a tool for process modelling and one of the oldest.

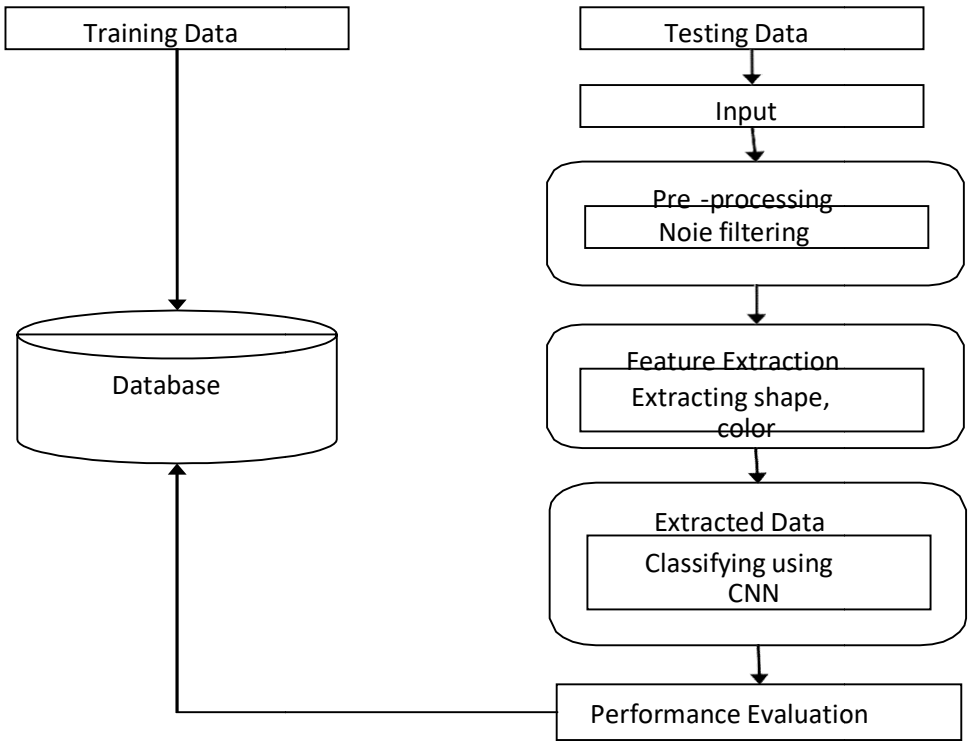


SOLUTION & TECHNICAL ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System

architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

13.1



USER STORIES

6. PROJECT PLANNING & SCHEDULING

Project Planning Phase

Date	18 November 2022
Team ID	PNT2022TMID35070
Project Name	Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies

SI. NO	MILESTONE	ACTIVITIES	DATE
1	Preparation Phase	Pre-requisites	24 Aug 2022
		Prior knowledge	25 Aug 2022
		Project Structure	23 Aug 2022
		Project Flow	23 Aug 2022
		Project Objectives	22 Aug 2022
		Registrations	26 Aug 2022
		Environment Set-up	27 Aug 2022
2	Ideation Phase	Literature Survey	29 Aug 2022- 03 Sept 2022
		Empathy Map	05 Sept 2022– 07 Sept 2022

		Problem Statement	08 Sept 2022 – 10 Sept 2022
		Brainstorming	12 Sept 2022 – 16 Sept 2022
3	Project Design Phase - I	Proposed Solution	19 Sept 2022 – 23 Sept 2022
		Problem Solution Fit	24 Sept 2022 – 28 Sept 2022
		Solution Architecture	29 Sept 2022 – 01 Oct 2022
4	Project Design Phase - II	Customer Journey	03 Oct 2022 – 08 Oct 2022
		Requirement Analysis	09 Oct 2022 – 11 Oct 2022
		Data Flow Diagrams	12 Oct 2022 – 13 Oct 2022
		Technology Architecture	14 Oct 2022 – 15 Oct 2022
5	Project Planning Phase	Milestones & Tasks	17 Oct 2022 – 18 Oct 2022
		Sprint Schedules	19 Oct 2022- 22 Oct 2022
6	Project Development Phase	Sprint-1	24 Oct 2022 – 29 Oct 2022
		Sprint-2	31 Oct 2022 –

			05 Nov 2022
		Sprint-3	07 Nov 2022 – 12 Nov 2022
		Sprint-4	14 Nov 2022 – 19 Nov 2022

SPRINT PLANNING & ESTIMATION

6.2 SPRINT DELIVERY SCHEDULE

SPRINT 1

IMAGE PRE PROCESSING

Body:

1. **IMPORT THE IMAGEDATAGENERATOR LIBRARY :**

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

2. **CONFIGURE IMAGEDATAGENERATOR CLASS IMAGE DATA AUGMENTATION :**

```
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.1, zoom_range = 0.1, horizontal_flip =
True) test_datagen = ImageDataGenerator(rescale =
1./255)
```

3. **APPLY IMAGEDATAGENERATOR FUNCTIONALITY TO TRAINSET AND TESTSET :**

```
training_set =
```

train_datagen.flow_from_directory('/content/drive/MyDrive/body/trainin
g',target_size = (224, 224),batch_size = 10,class_mode =
'categorical') test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/body/validati
on',target_size = (224, 224),batch_size = 10,class_mode = 'categorical')

Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.

Level:

1. **Import The ImageDataGenerator Library :** from tensorflow.keras.preprocessing.image import ImageDataGenerator

2. **Configure ImageDataGenerator Class :**

train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.1, zoom_range = 0.1, horizontal_flip =
True) test_datagen = ImageDataGenerator(rescale =
1./255)

3. **Apply ImageDataGenerator Functionality To Trainset And Testset :**

```
training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/level/traini
ng',target_size = (224, 224),batch_size = 10,class_mode =
'categorical') test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/level/validat
ion',target_size = (224, 224),batch_size = 10,class_mode = 'categorical')
```

Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.

SPRINT 2

MODEL BUILDING

Body:

1. Importing The Model Building Libraries

```
import tensorflow as tf from tensorflow.keras.layers import Input, Lambda,
Dense, Flatten from
tensorflow.keras.models import Model from
tensorflow.keras.applications.vgg16 import VGG16 from
tensorflow.keras.applications.vgg19 import VGG19 from
tensorflow.keras.preprocessing
import image from
tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img from
tensorflow.keras.models import Sequential import
numpy as
```

np from glob import glob

2. Loading The Model

```
IMAGE_SIZE = [224, 224] train_path = '/content/drive/MyDrive/body/training' valid_path = '/content/drive/MyDrive/body/validation' vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

Downloading data from https://storage.googleapis.com/tensorflow/kerasapplications/vgg16/vgg16_weights_tf_dim_ordering_ tf_kernels_notop.h5 58889256/58889256 [==============================] - 0s 0us/step

3. Adding Flatten Layer

```
for layer in vgg16.layers:layer.trainable = False
folders = glob('/content/drive/MyDrive/body/training/*')
folders
['/content/drive/MyDrive/body/training/02-side',
 '/content/drive/MyDrive/body/training/00-front',
'/content/drive/MyDrive/body/training/01-rear'] x = Flatten()(vgg16.output)
len(folders)
3
```

4. Adding Output Layer

```
prediction = Dense(len(folders), activation='softmax')(x)
```

5. Creating A Model Object

```
model = Model(inputs=vgg16.input, outputs=prediction) model.summary()
```

Model: "model"

```
_____
___ 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
block3_conv3 (Conv2D)       (None, 56, 56, 256)            590080
block3_pool (MaxPooling2D)  (None, 28, 28, 256)            0
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
```

flatten (Flatten) (None, 25088) 0
dense
(Dense) (None, 3) 75267

==========

==========

Total params: 14,789,955

Trainable params: 75,267

Non-trainable params: 14,714,688

_____

__

6. Configure The Learning Process

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

7. Train The Model

8. Save The Model

```
from tensorflow.keras.models import load_model
model.save('/content/drive/MyDrive/ibm project/Intelligent Vehicle
Damage Assessment & Cost Estimator/MODEL/BODY.h5')
```

9. Test The Model

```
from tensorflow.keras.models import load_model
import cv2 from skimage.transform import resize
```

```
model = load_model('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/BODY.h5')

def detect(frame):   img = cv2.resize(frame,(224,224))   img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
if(np.max(img)>1):      img = img/255.0      img = np.array([img])     prediction = model.predict(img)
label  =  ["front","rear","side"]              preds  =  label[np.argmax(prediction)]     return preds

data  =  "/content/drive/MyDrive/body/training/00-front/0007.JPEG"  image  =  cv2.imread(data)
print(detect(image))
```

1/1 [==============================] - 1s 812ms/step front

Level:

1. Importing The Model Building Libraries

```
import tensorflow as tf from tensorflow.keras.layers import Input, Lambda, Dense, Flatten from
tensorflow.keras.models import Model from tensorflow.keras.applications.vgg16 import VGG16 from
tensorflow.keras.applications.vgg19 import VGG19 from tensorflow.keras.preprocessing
```

```
import image from
tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img from
tensorflow.keras.models import Sequential import
numpy as
np from glob import glob
```

2. Loading The Model

```
IMAGE_SIZE = [224, 224] train_path =
'/content/drive/MyDrive/level/training' valid_path
= '/content/drive/MyDrive/level/validation'
```

3. Adding Flatten Layer vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False) Downloading data from https://storage.googleapis.com/tensorflow/kerasapplications/vgg16/vgg16_weights_tf_dim_ordering_ tf_kernels_notop.h5 58889256/58889256 [==============================] - 2s 0us/step

```
for layer in vgg16.layers:layer.trainable = False folders =
glob('/content/drive/MyDrive/level/training/*') folders

['/content/drive/MyDrive/level/training/02-moderate',
 '/content/drive/MyDrive/level/training/03-severe',
'/content/drive/MyDrive/level/training/01-minor'] x = Flatten()(vgg16.output)
len(folders)
3
```

4. Adding Output Layer

```
prediction = Dense(len(folders), activation='softmax')(x)
```

5. Creating A Model Object

```
model = Model(inputs=vgg16.input, outputs=prediction) model.summary()
```

Model: "model"

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
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
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

flatten (Flatten) (None, 25088) 0

dense

(Dense) (None, 3) 75267

===

==================== Total params: 14,789,955

Trainable params: 75,267

Non-trainable params: 14,714,688

_____

__

6. Configure The Learning Process

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

7. Train The Model

```
r = model.fit_generator( training_set,
validation_data=test_set, epochs=5,
```

steps_per_epoch=len(training_set),
validation_steps=len(test_set))

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

Epoch 1/5
98/98 [==============================] - 407s 4s/step - loss: 1.2409 - accuracy: 0.5628 - val_loss: 1.2019 - val_accuracy: 0.5614 Epoch 2/5
98/98 [==============================] - 18s 179ms/step - loss: 0.7316 - accuracy: 0.7191 - val_loss: 0.9586 - val_accuracy: 0.6082
Epoch 3/5
98/98 [==============================] - 16s 164ms/step - loss: 0.5469 - accuracy: 0.7957 - val_loss: 1.0207 - val_accuracy: 0.6140
Epoch 4/5
98/98 [==============================] - 16s 167ms/step - loss: 0.4278 - accuracy: 0.8223 - val_loss: 1.6515 - val_accuracy: 0.5965
Epoch 5/5
98/98 [==============================] - 17s 177ms/step - loss: 0.4449 - accuracy: 0.8284 - val_loss: 1.2299 - val_accuracy: 0.6199

8. Save The Model

```
from tensorflow.keras.models import load_model
model.save('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/LEVEL.h5')
```

9. Test The Model

```
from tensorflow.keras.models import load_model
import cv2 from skimage.transform import resize

model = load_model('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/LEVEL.h5')

def detect(frame):  img = cv2.resize(frame,(224,224))  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
if(np.max(img)>1):  img = img/255.0  img = np.array([img])  prediction = model.predict(img)  label = ["minor","moderate","severe"]  preds = label[np.argmax(prediction)]  return preds

data = "/content/drive/MyDrive/level/training/01-minor/0007.JPEG" image = cv2.imread(data)

print(detect(image))
```

1/1 [==============================] - 0s 157ms/step minor

SPRINT 3

SPRINT 4

It also consist of main body and main level codes

- **CODING & SOLUTIONING**

- **FEATURE 1**
- **7.2 FEATURE 2**
- **7.3 DATABASE SCHEMA**

8. TESTING

TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on "HOW" to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.

- Reusable: Can be reused if necessary.

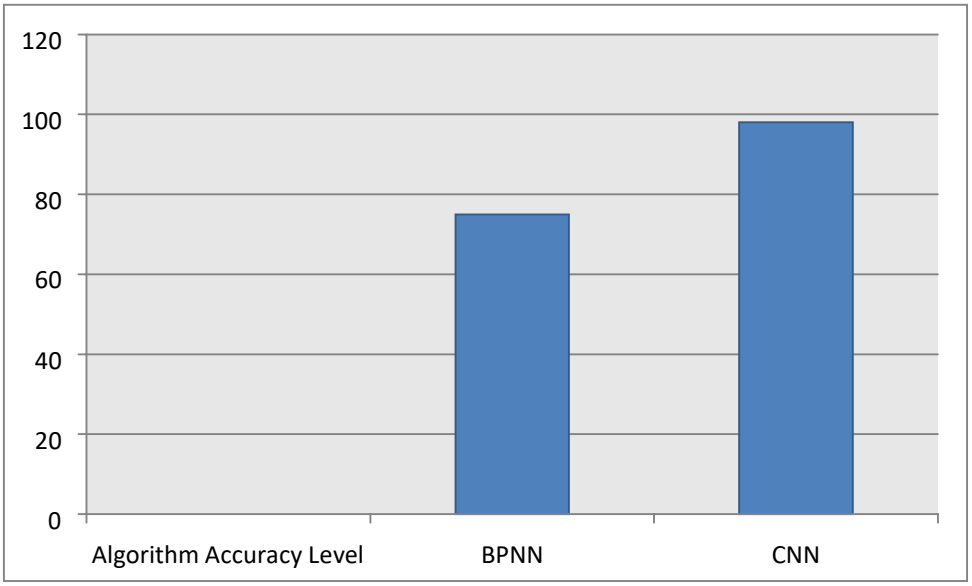
S.NO	Scenario	Input	Excepted output	Actual output
1	User login	User name and password	Login	Login success.
2	Upload Image	Upload damaged vehicle image as a input	Detecting object and analyze for claim insurance	Details are stored in a database.

8.2 USER ACCEPTANCE TESTING

This sort of testing is carried out by users, clients, or other authorised bodies to identify the requirements and operational procedures of an application or piece of software. The most crucial stage of testing is acceptance testing since it determines whether or not the customer will accept the application or programme. It could entail the application's U.I., performance, usability, and usefulness. It is also referred to as end-user testing, operational acceptance testing, and user acceptance testing (UAT).

9. RESULTS

PERFORMANCE METRICS



10. ADVANTAGES & DISADVANTAGES

ADVANTAGE

- Digitalized claim process makes easy to use
- Give the accurate result of the damaged vehicle
- Helps the insurance company to analyze the damaged vehicle and also payment process.

DISADVANTAGE

- It will take more time to claim the insurance in manual process
- Because of incorrect claims, the company behaves badly and doesn't make payments currently.
- Poor customer support

11. CONCLUSION

In this research proposal, a neural network-based solution for automobile detection will be used to address the issues of automotive damage analysis and position and severity prediction. This project does several tasks in one bundle. The method will unquestionably assist the insurance firms in conducting far more thorough and systematic analyses of the vehicle damage. Simply sending the system a photograph of the vehicle, it will evaluate it and determine whether there is damage of any type, where it is located, and how severe it is.

12. FUTURE SCOPE

In future work, need to use several regularisation methods with a big dataset in our next work. Anticipate the cost of a car damaged component more accurately and reliably if we have higher quality datasets that include the attributes of a car (make, model, and year of production), location data, kind of damaged part, and repair cost. This study makes it possible to work together on picture recognition projects in the future, with a focus on the auto insurance industry. The study was able to accurately validate the presence of damage, its location, and its degree while eliminating human bias. These can be further enhanced by adding the on the fly data augmentation approaches.

13. APPENDIX SOURCE CODE

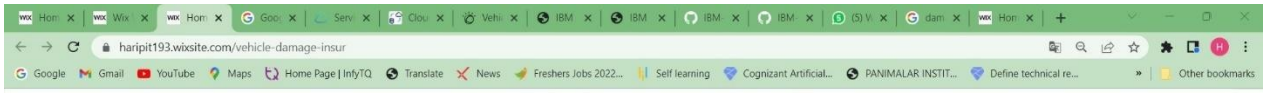
```
from flask import Flask, render_template, flash,
request,session from cloudant.client import Cloudant
import cv2 client = Cloudant.iam("eb55a2b7-ae45-
4df8-8d1c-69c5229ffdbe-
bluemix","YzG5FZg9Vs_HScOBZaWyVXm7PpNjbPrmPaPMfHx7w3X9",c
onnect=
```

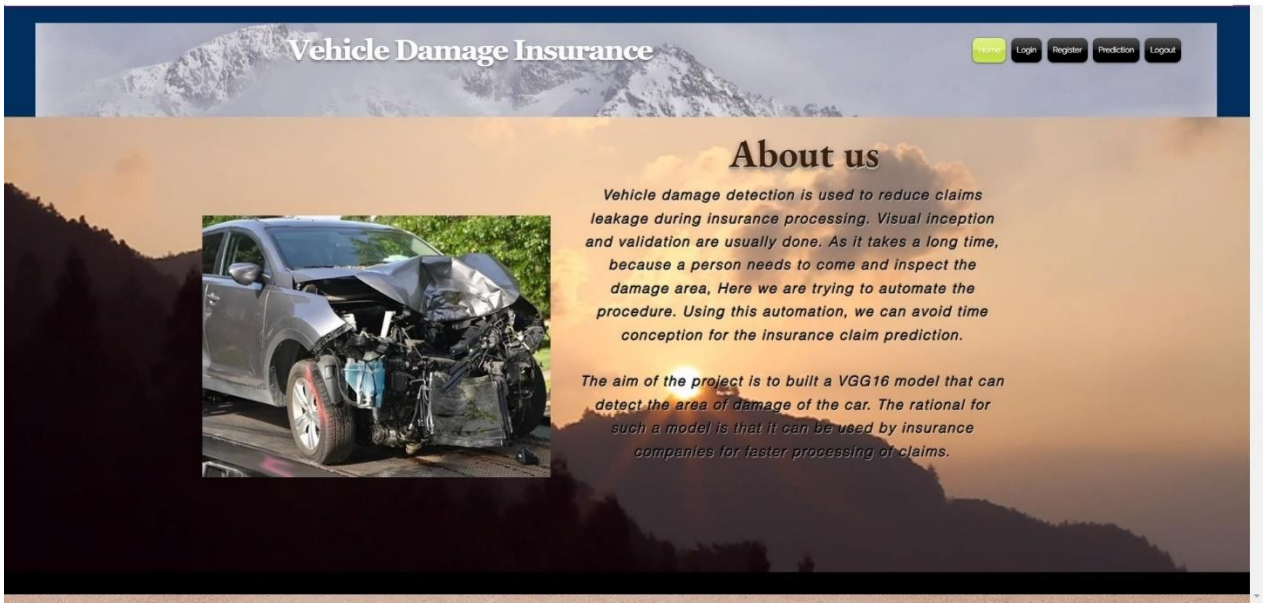
```
True) my_database = client.create_database("database-
dharan") app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] =
'7d441f27d441f27567d441f2b6176a'
@app.route("/")
def homepage():
    return render_template('index.html')
@app.route("/userhome"
) def userhome():
    return render_template('userhome.html')
@app.route("/addamoun
t")
@app.route("/NewUser"
) def NewUser():
    return render_template('NewUser.html')

@app.route("/user")
def user():

    return render_template('user.html')
@app.route("/newuse",methods=['GET','
POST']) def newuse():     if
request.method == 'POST':
        x = [x for x in
request.form.values()]         print(x)
data = {
            '_id': x[1],
            'name': x[0],
```

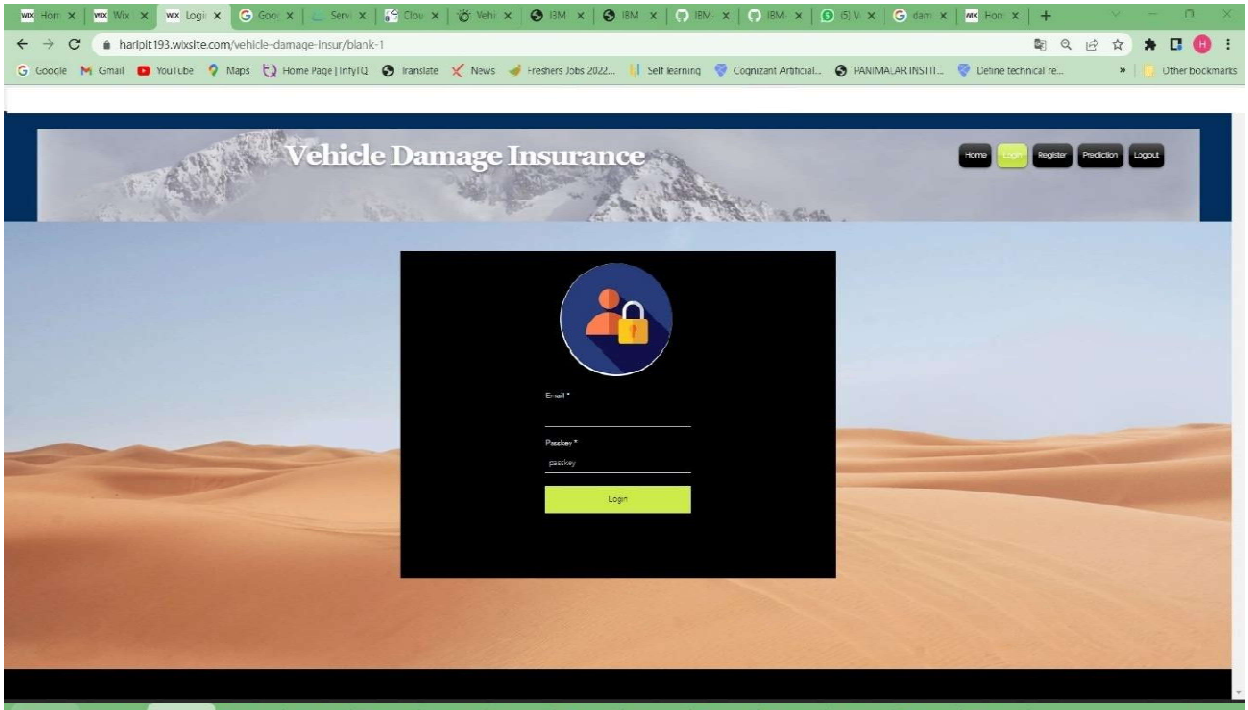
```
            'psw': x[2]
        }         print(data)         query = {'_id':
{'Seq': data['_id']}}         docs =
my_database.get_query_result(query)
print(docs)         print(len(docs.all()))
if (len(docs.all()) == 0):
            url = my_database.create_document(data)             return
render_template('goback.html', data="Register, please login using your
details")         else:
            return render_template('goback.html', data="You are already a
member, please login using your details")
@app.route("/userlog", methods=['GET',
'POST']) def userlog():         if
request.method == 'POST':             user =
request.form['_id']             passw =
request.form['psw']             print(user,
passw)             query = {'_id': {'$eq': user}}
docs =
my_database.get_query_result(query)
print(docs)             print(len(docs.all()))
if (len(docs.all()) == 0):
                return render_template('goback.html', pred="The username is
not found.")             else:                 if ((user == docs[0][0]['_id'] and
passw == docs[0][0]['psw'])):
                    return
render_template("userhome.html")                 else:
                    return    render_template('goback.html',data="user    name    and
password incorrect")
```

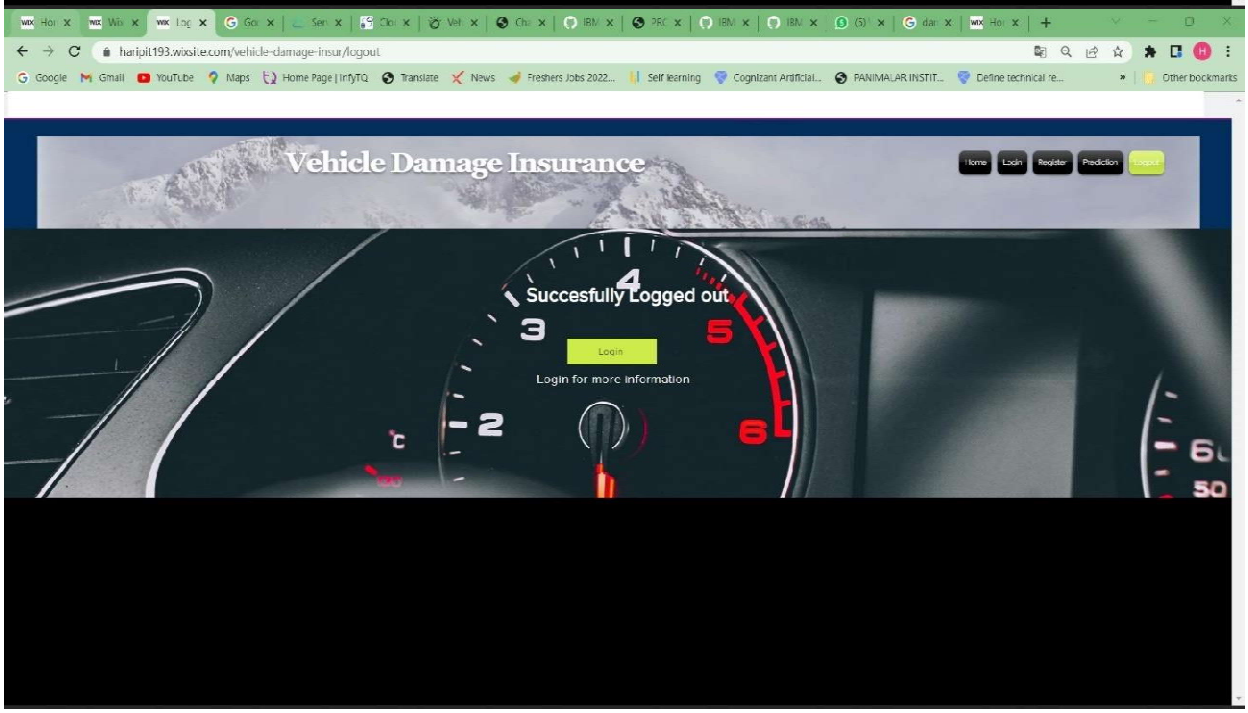
```
@app.route("/predict", methods=['GET', 'POST']) def predict():     if
request.method == 'POST':         file = request.files['fileupload']
file.save('static/Out/Test.jpg')         import warnings
warnings.filterwarnings('ignore')     import tensorflow as tf
classifierLoad = tf.keras.models.load_model('body.h5')         import
numpy as np         from keras.preprocessing import image
test_image = image.load_img('static/Out/Test.jpg', target_size=(200,
200))         img1 = cv2.imread('static/Out/Test.jpg')         #
test_image = image.img_to_array(test_image)         test_image =
np.expand_dims(test_image, axis=0)         result =
classifierLoad.predict(test_image)         result1 = ''         if result[0][0]
== 1:             result1 = "front"         elif result[0][1] == 1:
result1 = "rear"         elif result[0][2] == 1:             result1 = "side"
file = request.files['fileupload1']         file.save('static/Out/Test1.jpg')
import warnings         warnings.filterwarnings('ignore')         import
tensorflow as tf         classifierLoad =
tf.keras.models.load_model('level.h5')         import numpy as np
from keras.preprocessing import image         test_image =
image.load_img('static/Out/Test1.jpg', target_size=(200, 200))
img1 = cv2.imread('static/Out/Test1.jpg')         # test_image =
image.img_to_array(test_image)
        test_image = np.expand_dims(test_image,
axis=0)         result =
classifierLoad.predict(test_image)         result2
= ''         if result[0][0] == 1:             result2 =
"minor"         elif result[0][1] == 1:
result2 = "moderate"         elif result[0][2] ==
1:             result2 = "severe"         if (result1 ==
"front" and result2 == "minor"):
```

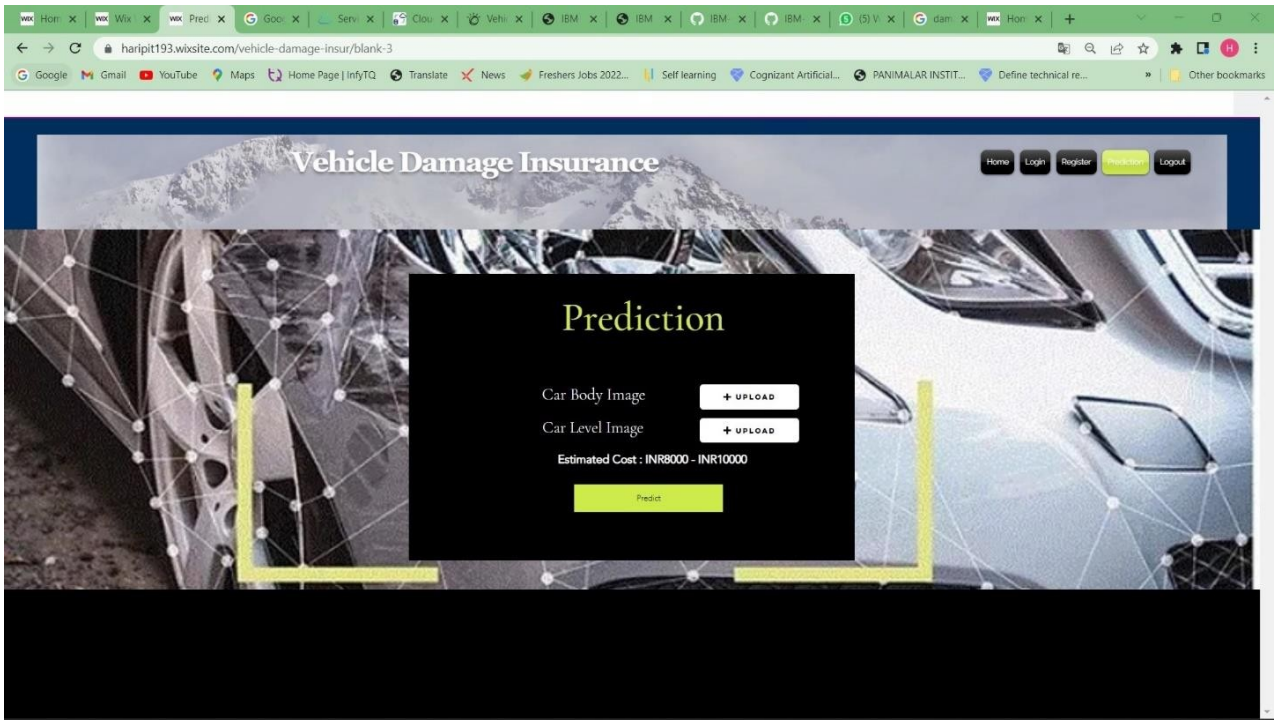
```
            value = "3000 - 5000 INR"            elif
        (result1 == "front" and result2 == "moderate"):
            value = "6000 8000 INR"            elif
        (result1 == "front" and result2 == "severe"):
            value = "9000 11000 INR"          elif
        (result1 == "rear" and result2 == "minor"):
            value = "4000 - 6000 INR"            elif
        (result1 == "rear" and result2 == "moderate"):
            value = "7000 9000 INR"           elif
        (result1 == "rear" and result2 == "severe"):
            value = "11000 - 13000 INR"       elif
        (result1 == "side" and result2 == "minor"):
            value = "6000 - 8000 INR"            elif
        (result1 == "side" and result2 == "moderate"):
            value = "9000 - 11000 INR"        elif
        (result1 == "side" and result2 == "severe"):
            value = "12000 - 15000
        INR"        else:
            value = "16000 - 50000 INR"        return
        render_template('userhome.html', prediction=value) if
        __name__== '__main__':
app.run(debug=True, use_reloader=True)
```


haripit193.wixsite.com/vehicle-damage-insur
Google
Gmail
YouTube
Maps
Home Page | InfyTQ
Translate
News
Freshers Jobs 2022...
Self learning
Cognizant Artificial...
PANIMALAR INSTIT...
Define technical re...
Other bookmarks


Vehicle Damage Insurance
Home
Login
Register
Prediction
Logout
About us
Vehicle damage detection is used to reduce claims leakage during insurance processing. Visual inception and validation are usually done. As it takes a long time, because a person needs to come and inspect the damage area, Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim prediction.
The aim of the project is to built a VGG16 model that can detect the area of damage of the car. The rational for such a model is that it can be used by insurance companies for faster processing of claims.


haripit193.wixsite.com/vehicle-damage-insur/blank-1
Vehicle Damage Insurance
Home
Login
Register
Prediction
Logout
E-mail *
Passkey *
passkey
Login


haripit193.wixsite.com/vehicle-damage-insur/logout
Vehicle Damage Insurance
Home
Login
Register
Prediction
Logout
Succesfully Logged out
Login
Login for more information



GITHUB & PROJECT DEMO LINK

[IBM-EPBL/IBM-Project-44564-1660725272]