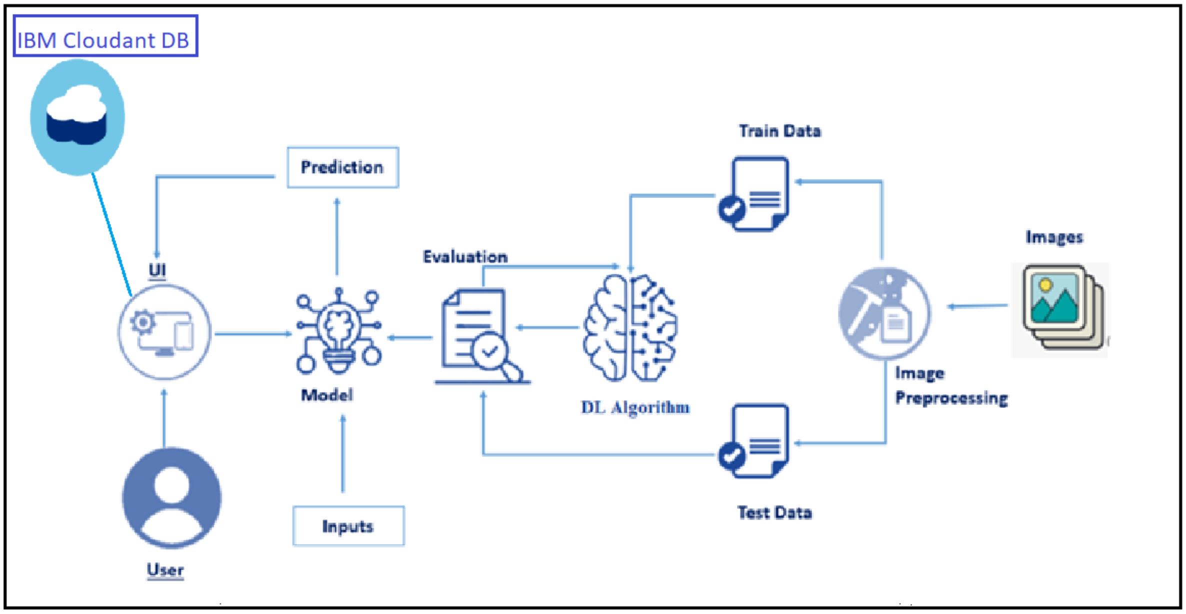
1.INTRODUCTION

1.1 Overview

Nowadays, 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.

The aim of this project is to build a VGG16 model that can detect the area of damage on a car. The rationale for such a model is that it can be used by insurance companies for faster processing of claims if users can upload pics and the model can assess damage(be it dent scratch from and estimates the cost of damage. This model can also be used by lenders if they are underwriting a car loan especially for a used car.



1.2 Purpose:

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.

Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim procedure.

2 LITERATURE SURVEY:

2.1 Existing problem:

The insurance industry is one of the first industries invested in innovation, the latest technology and artificial intelligence (AI) . In today’s world, when the rate of car accidents is increasing, car insurance companies waste millions of dollars annually, due to claims leakage. The sense of AI technology based on machine learning and deep learning can help problems such as analyzing and processing data, detecting frauds, lessening risks and automating claim process in insurance industries . So, insurance firms have looked for faster damage assessment and agreement of claims.

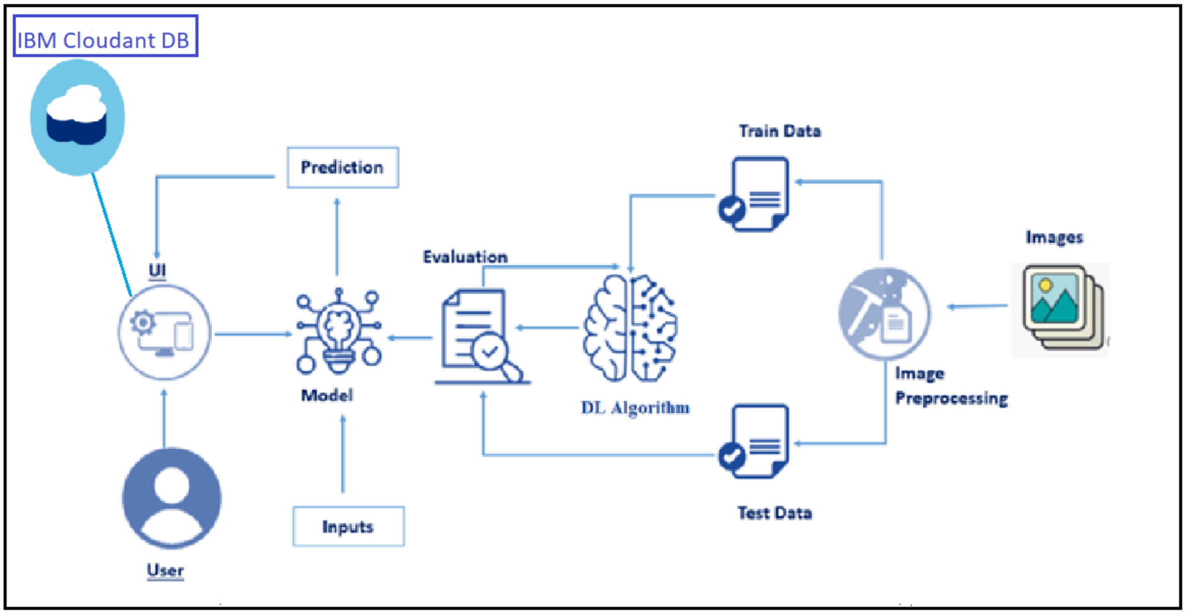
2.2 Proposed solution:

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. Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim procedure.

The user interacts with the UI (User Interface) to choose the image.The chosen image is analyzed by the model which is integrated with the flask application.VGG16 Model analyzes the image, then the prediction is showcased on the Flask UI.

3 THEORITICAL ANALYSIS:

3.1 Block diagram:



3.2 Hardware / Software designing:

- Jupyter Notebook
- Sypder

•HTML-Hyper Text Markup Language

•Flask

-Python Framework

•CAR IMAGES DATA SET

4 EXPERIMENTAL INVESTIGATIONS:

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. Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim procedure.

- The user interacts with the UI (User Interface) to choose the image.The chosen image is analyzed by the model which is integrated with the flask application.VGG16 Model analyzes the image, then the prediction is showcased on the Flask UI.

5 FLOWCHART:

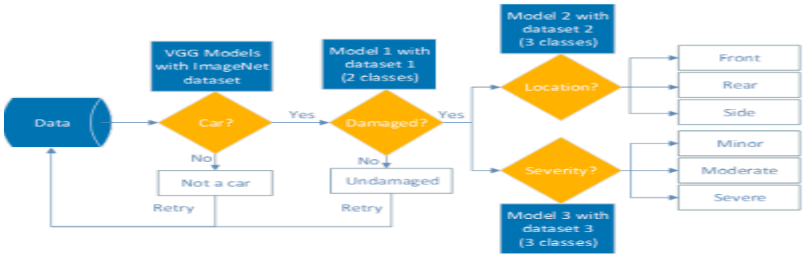
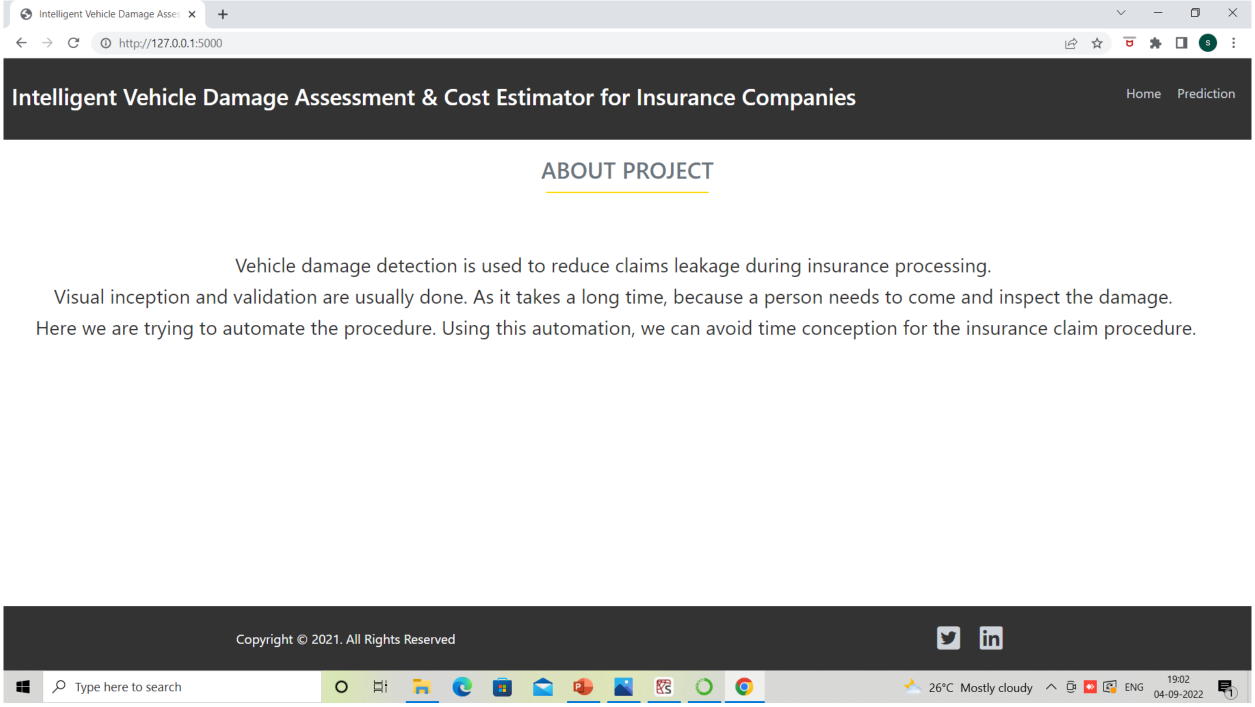

Model 2 with dataset 2 (3 classes)
VGG Models with ImageNet dataset
Model 1 with dataset 1 (2 classes)
Data
Car?
Yes
Damaged?
Yes
Location?
Front
Rear
Side
No
Not a car
No
Undamaged
Retry
Retry
Severity?
Minor
Moderate
Severe
Model 3 with dataset 3 (3 classes)

Figure 1: A flow chart of developing car damage assessment pipelines

6 RESULT:

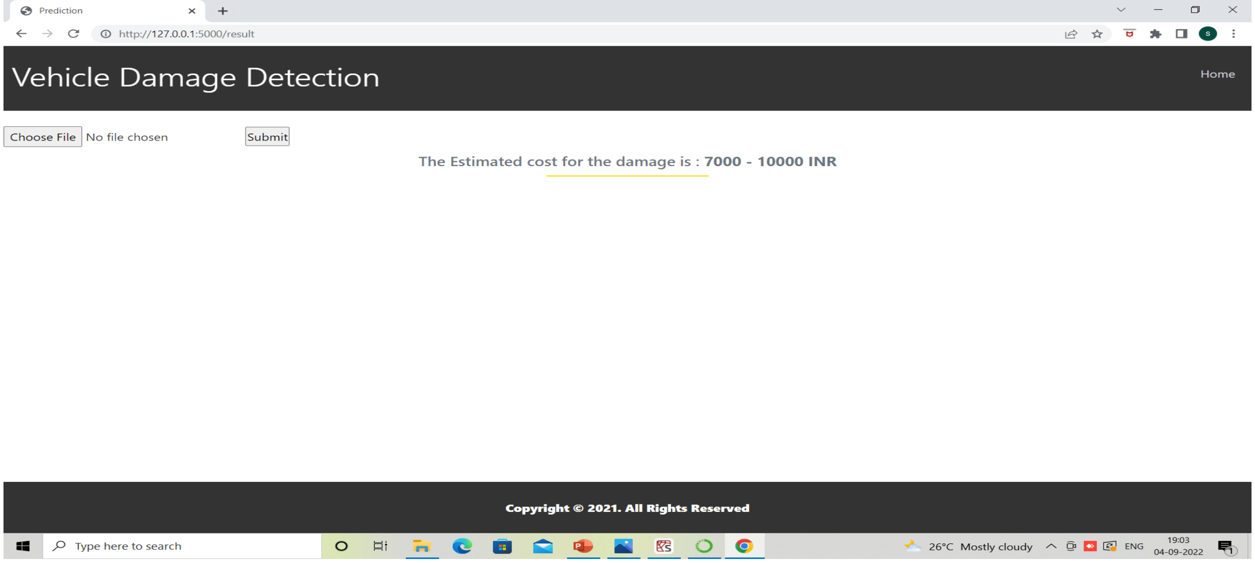


Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

ABOUT PROJECT

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.
Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim procedure.

Copyright © 2021. All Rights Reserved



Vehicle Damage Detection

The Estimated cost for the damage is : **7000 - 10000 INR**

Copyright © 2021. All Rights Reserved

7 ADVANTAGES & DISADVANTAGES:

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. Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim procedure.

we got only 98 % accuracy for this model.any model doesnt give 100% accuracy.

Regarding our proposed models, we still face the overfitting problem in our models. Thus, in future work, we will utilize other types of regularization techniques and other pre-trained CNN models with a large dataset to fit that problem. If we have higher quality datasets, including the features of a car (make, model and the year of manufacture), location information, type of damaged part and repair cost, we can predict the cost of a car damaged part to be more reliable and accurate.

8 APPLICATIONS:

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. Here we are trying to automate the procedure. Using this automation, we can avoid time conception for the insurance claim procedure.

9 CONCLUSION :

We described applicable deep learning-based algorithms for car damage assessment. We created new datasets when there is regularization technique to fit our specific tasks. We observed that training with a small dataset is not sufficient to get the best accuracy based on deep learning approach. In addition to this, it was not enough just using L2 regularization technique in our system. After analyzing our models, we find out that the results of using transfer learning and regularization can work better than those of fine-tuning. After that, the performances of VGG19 are better than VGG16. All of the above, our pre-trained VGG models not only detect damaged part of a car but also assess its location and severity...

10 FUTURE SCOPE:

Regarding our proposed models, we still face the overfitting problem in our models. Thus, in future work, we will utilize other types of regularization techniques and other pre-trained CNN models with a large dataset to fit that problem. If we have higher quality datasets, including the features of a car (make, model and the year of manufacture), location information, type of damaged part and repair cost, we can predict the cost of a car damaged part to be more reliable and accurate.

11 BIBILOGRAPHY:

[1] Najmeddine Dhieb, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. 2019. Extreme Gradient Boosting MachineLearning Algorithm For Safe Auto Insurance Operations.
In2019 IEEE International Conference of Vehicular Electronicsand Safety (ICVES). IEEE, 1–5.
[2] Najmeddine Dhieb, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. 2019.
A very deep transfer learning modelfor vehicle damage detection and localization.

In2019 31st International Conference on Microelectronics (ICM).

[3] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neuralnetworks?. InAdvances in neural information processing systems. 3320–3328.

[4] Mahavir Dwivedi, Malik Hashmat Shadab, SN Omkar, Edgar Bosco Monis, Bharat Khanna, and Satya Ranjan. [n.d.].Deep Learning Based Car Damage Classification and Detection. ([n. d.] [5] Kalpesh Patil, Mandar Kulkarni, Anand Sriraman, and Shirish Karande. 2017. Deep learning based car damageclassification. In2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 50–54.

[6] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning.Journal ofBig Data6, 1 (2019), 60.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical imagedatabase. In2009 IEEE conference on computer vision and pattern recognition. IEEE, 248–255.

[8] Jeffrey de Deijn. 2018. Automatic Car Damage Recognition using Convolutional Neural Networks. (2018).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neuralnetworks. InAdvances in neural information processing systems. 1097–1105.

[11] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition.arXiv preprint arXiv:1409.1556(2014).

[12] Srikanth Tammina. [n.d.]. Transfer learning using VGG-16 with Deep Convolutional Neural Network for ClassifyingImages. ([n. d.]).

[13] Syukron Abu Ishaq Alfarozi, Kitsuchart Pasupa, Hiromichi Hashizume, Kuntpong Woraratpanya, and MasanoriSugimoto. 2019. Robust and Unified VLC Decoding System for Square Wave Quadrature Amplitude Modulation UsingDeep Learning Approach.IEEE Access7 (2019).

[14] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning.IEEE

Transactions on knowledge and dataengineering22, 10 (2009), 1345–1359.
[15] Ranjodh Singh, Meghna P Ayyar, Tata Sri Pavan, Sandeep Gosain, and Rajiv Ratn Shah. 2019. Automating Car InsuranceClaims Using Deep Learning Techniques. In2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM).IEEE, 199–207.
[16] Pei Li, Bingyu Shen, and Weishan Dong. 2018. An anti-fraud system for car insurance claim based on visual evidence.arXiv preprint arXiv:1804.11207(2018)

APPENDIX

A. Source Code

app.py:

```
import re
import numpy as np
import os
from flask import Flask, app,request,render_template
from tensorflow.keras import models
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.python.ops.gen_array_ops import concat
from tensorflow.keras.applications.inception_v3 import preprocess_input
import requests
from flask import Flask, request, render_template, redirect, url_for
#Loading the model

model1=load_model("body.h5")
model2=load_model("level.h5")

app=Flask(__name__)

#default home page or route
@app.route('/')
```

```
def index():
    return render_template('index.html')



@app.route('/index')
def home():
    return render_template("index.html")



@app.route('/prediction')
def prediction():
    return render_template('prediction.html')


@app.route('/result',methods=["GET","POST"])
def res():
    if request.method=="POST":
        f=request.files['image']
        basepath=os.path.dirname(__file__) #getting the current path i.e where app.py is present
        #print("current path",basepath)
        filepath=os.path.join(basepath,'uploads',f.filename) #from anywhere in the system we can
give image but we want that image later  to process so we are saving it to uploads folder for
reusing
        #print("upload folder is",filepath)
        f.save(filepath)

        img=image.load_img(filepath,target_size=(224,224))
        x=image.img_to_array(img)#img to array
        x=np.expand_dims(x,axis=0)#used for adding one more dimension
        #print(x)
        img_data=preprocess_input(x)
        print(model1.predict(img_data), model2.predict(img_data))
        prediction1=np.argmax(model1.predict(img_data))
        prediction2=np.argmax(model2.predict(img_data))
        print(prediction1, prediction2)
        #prediction=model.predict(x)#instead of predict_classes(x) we can use predict(X) ----
>predict_classes(x) gave error
        #print("prediction is ",prediction)
        index1=['front', 'rear', 'side']
        index2=['minor', 'moderate', 'severe']
        #result = str(index[output[0]])
        result1 = index1[prediction1]
        result2 = index2[prediction2]
        print(result1,result2)
        if(result1 == "front" and result2 == "minor"):
          number = "3000 - 5000 INR"

        elif(result1 == "front" and result2 == "moderate"):
          number = "6000 - 8000 INR"

        elif(result1 == "front" and result2 == "severe"):
          number = "9000 - 11000 INR"

        elif(result1 == "rear" and result2 == "minor"):
          number = "4000 - 6000 INR"

        elif(result1 == "rear" and result2 == "moderate"):
          number = "7000 - 10000 INR"

        elif(result1 == "rear" and result2 == "severe"):
          number = "11000 - 13000 INR"

        elif(result1 == "side" and result2 == "minor"):
           number = "6000 - 8000 INR"

        elif(result1 == "side" and result2 == "moderate"):
           number = "9000 - 11000 INR"

        elif(result1 == "side" and result2 == "severe"):
           number = "12000 - 15000 INR"

        else:
          number = "15000 - 50000 INR"

        return render_template('prediction.html', prediction=number)



""" Running our application """
```

```
if __name__ == "__main__":
   app.run(debug = False)
```

prediction.html:

```
import re
import numpy as np
import os
from flask import Flask, app,request,render_template
from tensorflow.keras import models
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.python.ops.gen_array_ops import concat
from tensorflow.keras.applications.inception_v3 import preprocess_input
import requests
from flask import Flask, request, render_template, redirect, url_for
#Loading the model

model1=load_model("body.h5")
model2=load_model("level.h5")

app=Flask(__name__)

#default home page or route
@app.route('/')
def index():
   return render_template('index.html')



@app.route('/index')
def home():
   return render_template("index.html")



@app.route('/prediction')
def prediction():
    return render_template('prediction.html')


@app.route('/result',methods=["GET","POST"])
def res():
    if request.method=="POST":
        f=request.files['image']
        basepath=os.path.dirname(__file__) #getting the current path i.e where app.py is present
        #print("current path",basepath)
        filepath=os.path.join(basepath,'uploads',f.filename) #from anywhere in the system we can
give image but we want that image later  to process so we are saving it to uploads folder for
reusing
        #print("upload folder is",filepath)
        f.save(filepath)

        img=image.load_img(filepath,target_size=(224,224))
        x=image.img_to_array(img)#img to array
        x=np.expand_dims(x,axis=0)#used for adding one more dimension
        #print(x)
        img_data=preprocess_input(x)
        print(model1.predict(img_data), model2.predict(img_data))
        prediction1=np.argmax(model1.predict(img_data))
        prediction2=np.argmax(model2.predict(img_data))
        print(prediction1, prediction2)
        #prediction=model.predict(x)#instead of predict_classes(x) we can use predict(X) ----
>predict_classes(x) gave error
        #print("prediction is ",prediction)
        index1=['front', 'rear', 'side']
        index2=['minor', 'moderate', 'severe']
        #result = str(index[output[0]])
        result1 = index1[prediction1]
        result2 = index2[prediction2]
        print(result1,result2)
        if(result1 == "front" and result2 == "minor"):
            number = "3000 - 5000 INR"

        elif(result1 == "front" and result2 == "moderate"):
            number = "6000 - 8000 INR"

        elif(result1 == "front" and result2 == "severe"):
            number = "9000 - 11000 INR"
        elif(result1 == "rear" and result2 == "minor"):
            number = "4000 - 6000 INR"

        elif(result1 == "rear" and result2 == "moderate"):
            number = "7000 - 10000 INR"

        elif(result1 == "rear" and result2 == "severe"):
            number = "11000 - 13000 INR"

        elif(result1 == "side" and result2 == "minor"):
             number = "6000 - 8000 INR"

        elif(result1 == "side" and result2 == "moderate"):
             number = "9000 - 11000 INR"

        elif(result1 == "side" and result2 == "severe"):
             number = "12000 - 15000 INR"

        else:
            number = "15000 - 50000 INR"

        return render_template('prediction.html', prediction=number)



""" Running our application """
if __name__ == "__main__":
    app.run(debug = False)
```