Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies

A PROJECT REPORT

TEAM ID - PNT2022TMID01378

Submitted by

SANJAY. V	***211419205143***
SANJAY PREETH. D	***211419205144***
SATHISH KUMAR. G	***211419205149***
DHIVYESH ANAND. K.P	***211419205042***

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

TABLE OF FIGURES

Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies

DEPARTMENT OF INFORMATION TECHNOLOGY PANIMALAR ENGINEERING COLLEGE, CHENNAi-6000123

sanjavedha@gmail.com, sathishgopinath69@gmail.com, sanjaypreeth2000@gmail.com, dhivyeshanand123@gmail.com

Abstract: Due to leaking claims, a lot of money is currently being lost in the auto insurance industry. Underwriting leakage is defined as the difference between the amount actually paid for claims and the amount that would have been paid had all of the top industry procedures been followed. Testing and visual inspection have both been used to arrive at these conclusions. However there is no go scheme about for the estimation of the cost of the damage that has been dealt with the car. The project's goal is to create a VGG16 model that can identify a car's damage area. If users provide photos, the model can analyze the damage (be it a dent or scratch from an object), and it can also estimate the amount of damage. This allows insurance firms to process claims more quickly.

1. INTRODUCTION

Vehicles are significantly rising in today’s globe. Because there are more cars on the road, accidents happen more frequently because people 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.The company acts badly and currently doesn't make payments as a result of false accusations. This occurs as a result of claims leakage, which is the discrepancy between the sums secured by the firm and the sums that company should have secured in accordance with the claims. Even if the car's damage is easily seen, the claim process 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 system 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 an image 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.

1.1 PROJECT OVERVIEW

The main aim of the project is to provide the estimation and the level of the damage in their car through the AI techniques that have been proposed in the system. The system can utilizes machine learning approach as well as computer vision to decide the damage analysis, location of the damage as well as severity of the damage.

1.2 PURPOSE

The purpose of the model is mainly to provide a neural network-based solution for car detection which will be used to address the issues of automotive damage analysis and position and severity prediction. This project performs numerous tasks in one bundle. The method will unquestionably assist the insurance firms in conducting much 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 any damage has occurred, where it has occurred, and how severe it is.

2. LITERATURE SURVEY

2.1. Existing problem

The claim settlement team finds it difficult to process the claim application if there is lack of proper documentation. The insurance claim company mainly aims to increase the cost of the damage that has been made. As the customer has a least knowledge regarding the cost and the level of damage, this is a main advantage to the insurance companies.

2.2. References

TITLE	**AUTHOR**	**FINDINGS**	**ADVANTAGES**
Deep Learning-Based Car Damage Classification and Detection – **Year** - 2019	Mahavir Dwivedi, Hashmat Shadab Malik, S. N. Omkar, Edgar Bosco Monis, Bharat Khanna, Satya Ranjan Samal, Ayush Tiwari & Aditya Rathi	In this paper, they have worked on the problem of vehicle damage classification/detection which can be used by insurance companies to automate the process of vehicle insurance claims in a quick fashion. The recent advances in computer vision largely due to the adoption of fast, scalable and end-to-end trainable convolutional neural networks make it technically feasible to recognize vehicle damages using deep convolutional networks. They manually collected and annotated images from various online sources containing different types of vehicle damages.	Using CNN models pre-trained on ImageNet dataset and using several other techniques to improve the performance of the system, they theyre able to achieve top accuracy of 96.39%, significantly better than the current results in this work. Furthermore, to detect the region of damage, they used state-of-the-art YOLO

Who Is Liable When a Driverless Car Crashes? **Year** - 2021	Muhammad Uzair	This work discusses all those aspects which should be considered in order to find out who is liable, i.e., an operator, owner, manufacturer, government entity, software provider, network provider, original equipment manufacturer (OEM), etc., as traditional tort rules will not help to find out the liability in case of an AV accident.	The work comprehensively discusses different liabilities ranging from legal, civil, operator, criminal, moral, product, insurance, etc., to find out who is liable in case of an AV accident, as compared to the existing literature which generally discusses one or two aspects only.
Improving Automobile Insurance Claims Frequency Prediction with Telematics Car Driving data **Year** - 2021	Shengwang Meng, He Wang, Yanlin Shi and Guangyuan Gao	In this paper, with automobile insurance claims data and associated telematics car driving data, they propose a supervised driving risk scoring neural network model. This one-dimensional convolutional neural network takes time series of individual car driving trips as input and returns a risk score in the unit range of (0,1).	After compared with non-telematics-based insurers, telematics-based insurers can discover more heterogeneity in their portfolio and attract safer drivers with premiums discounts.
Machine Learning Approaches for Auto Insurance Big Data **Year** - 2021	Mohamed Hanafy and Ruixing Ming	This study considers how automotive insurance providers incorporate machinery learning in their company, and explores how ML models can apply to insurance big data. They have utilized various ML methods, such as logistic regression, XGBoost, random forest, decision trees, naïve Bayes, and K-NN, to predict claim	This paper's main objective is to create an ML algorithm that accurately predicts claims occurrence. Thus, the model must effectively consider consumer details, such as the type of vehicle or the car's cost, that differ among the

		occurrence. Furthermore, they evaluate and compare these models' performances.	clients. The model's results (and provided that the forecast is accurate) confirmed that car insurance firms will make insurance coverage more available to more clients.
Assessing Car Damage with Convolutional Neural Networks **Year** - 2021	Harit Bandi; Suyash Joshi; Siddhant Bhagat; Amol Deshpande	This paper deals with estimating car damage, primarily with auto insurers as our key potential customers. For this purpose, three distinct Transfer Learning approaches are used which detect the presence of damage, location, and severity of the damage. The basis for algorithms used lies in Convolutional Neural Networks, customized to optimize accuracy.	This research fine-tunes a number of existing approaches and opens doors for collaboration in image recognition, particularly for the car insurance domain.
Regional Analysis Of Car Insurance In Montenegro Using Deep Learning Methods **Year** - 2021	Vladimir Kašćelan, Ljiljana Kašćelan, Milijana Novović Burić	This paper analyzes the regional distribution of the number of policies, total premiums and claims of 19486 legal entities as car insurance customers of one of the largest non-life insurance companies in Montenegro. Insureds are clustered using the k-means methods in four clusters: Poor-low risk, Middlelow risk, Theyalthy-middle risk and Luxury-high risk. The clusters are described using the Decision Tree (DT) models.	The research results are of particular importance to the insurance company, i.e. its marketing and actuary departments, in order to establish better communication, to develop more successful marketing strategies and to better assess the risk by region for individual groups of policyholders of similar characteristics.

Business Analytics and Car Insurance Mediation: a data driven model proposal **Year** - 2021	José Gonçalves Santos; Bernardo Abreu; Jorge Bernardino; Isabel Pedrosa	This work allows companies to make more reality-oriented and data-based models. They suggest a "pay-as-you-drive" model, which allows car insurance companies to submit proposals based on a larger number of variables and adjusted to each insured person, a model built with data collection by insurance companies, in order to adjust the insurance premium.	The contribution of this work corresponds to a new model that balances the insurance premium with the probability of unwanted occurrences.
Erie Insurance: Monitoring technology in the car insurance market and the issue of data privacy **Year** - 2021	Barbara A Manko	The influence of big data affects all aspects of life, including insurance. With car insurance, companies have the ability to go beyond the traditional driving history and current usage questions to customize policies for each holder, especially with the introduction of monitoring technology that is either installed directly into the car or accessed via cell phone.	The car insurance companies must be aware of the ethics of their actions and work to protect user data. Erie Insurance Company is specifically analysed in this article as an example.
Assessing the Performance of Random Forests for Modelling Claim Severity in Collision Car Insurance **Year** - 2021	Yves Staudt, ORCID and Joël Wagner	In this paper, they focus on the claim severity. First, they build two reference models, a generalized linear model and a generalized additive model, relying on a log-normal distribution of the severity and including the most significant factors. Thereby, they relate the continuous variables to the response in a nonlinear way. In the second step, they tune two random forest models, one for	Random forests with a log-normal transformation are the favourite choice for explaining right-sketheyd claims. Finally, when considering all indicators, they conclude that the generalized additive model has the best overall performance.

		the claim severity and one for the log-transformed claim severity, where the latter requires a transformation of the predicted results. They compare the prediction performance of the different models using the relative error, the root mean squared error and the goodness-of-lift statistics in combination with goodness-of-fit statistics.	
Deep Learning Based Car Damage Detection, Classification and Severity **Year** - 2021	Ritik Gandhi	In this paper they are using various pre-trained models such as VGG 16, VGG 19,Resnet50 and Densenet and based on these models, selecting the best performing models. They initially check whether the car is damaged or not using the Resnet50 model and if it's a damaged one they use the WPOD-net model to detect the license plate. To identify the damaged region, they use the YOLO model. At last, comes the damage severity which is implemented using the Densenet model .	From the above models they can safely conclude that Resnet model works best to detect whether a car is damaged or not, YOLO models to identify the car damage classification and the densenet model to check the severity of the car damage. Regarding the proposed models there are still overfitting issues but there is still room for improvements in terms of accuracy

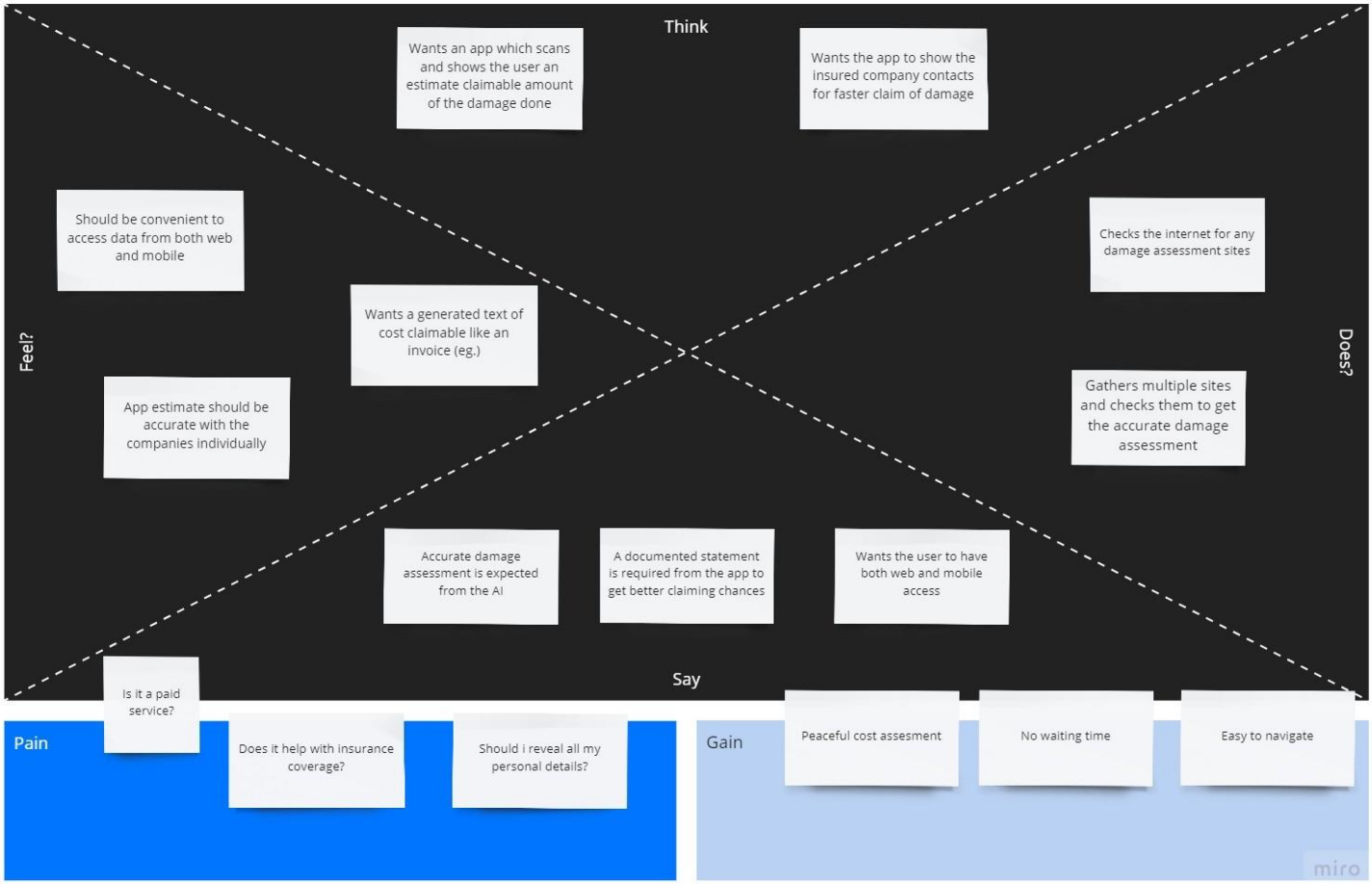
2.3. Problem Statement Definition

Our solution to this problem is to create a site which helps the user to get an estimate of claimable insurance amount for the damaged vehicle and measure the level of damage. With the help of vgg16 and certain python libraries we will be able to offer a solution for the given problem statement.

3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas

The Empathy Map is a collaborative tool that allows teams to gain deeper insight into their customers. Similar to user personas, empathy maps can represent user groups such as customer segments**.** The below empathy map gives the various user request and their ideas, problems, etc. they are facing in real time.



3.2. Ideation and Brainstorming

During the Ideation stage, design thinkers generate ideas by participating in imaginative and inquisitive exercises like Brainstorms and Worst Possible Idea. The below picture depicts the possible ideation and brainstorming ideas created by our members.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Customer	Find the exact cost of the damage in my car	I don't have enough storage space in my mobile	I have limited storage space in my mobile and I can't install any further applications	Uncomfortable
PS-2	Owner	To determine the severity of the damage in my car	I can't find proper and correct estimation for the damage	Lack of accurate information on the internet	Irritated

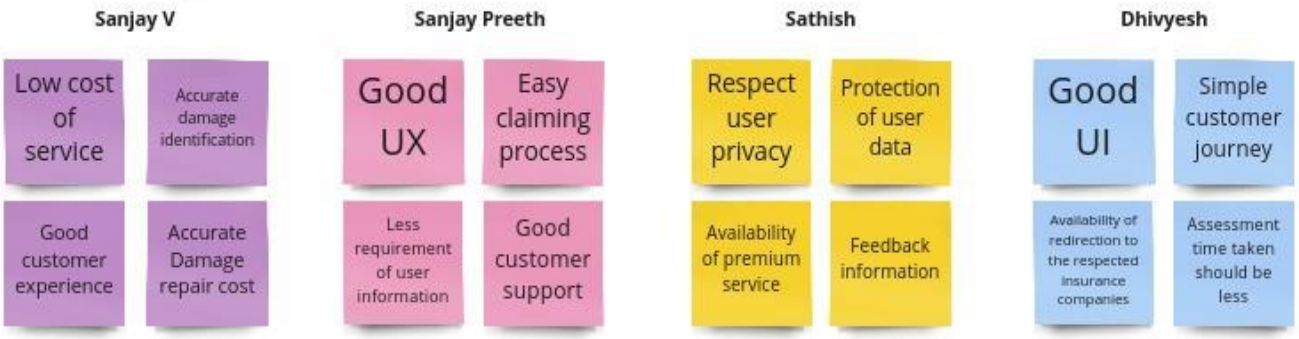
Problem Statement:

To find the claimable insurance amount for the damaged vehicles

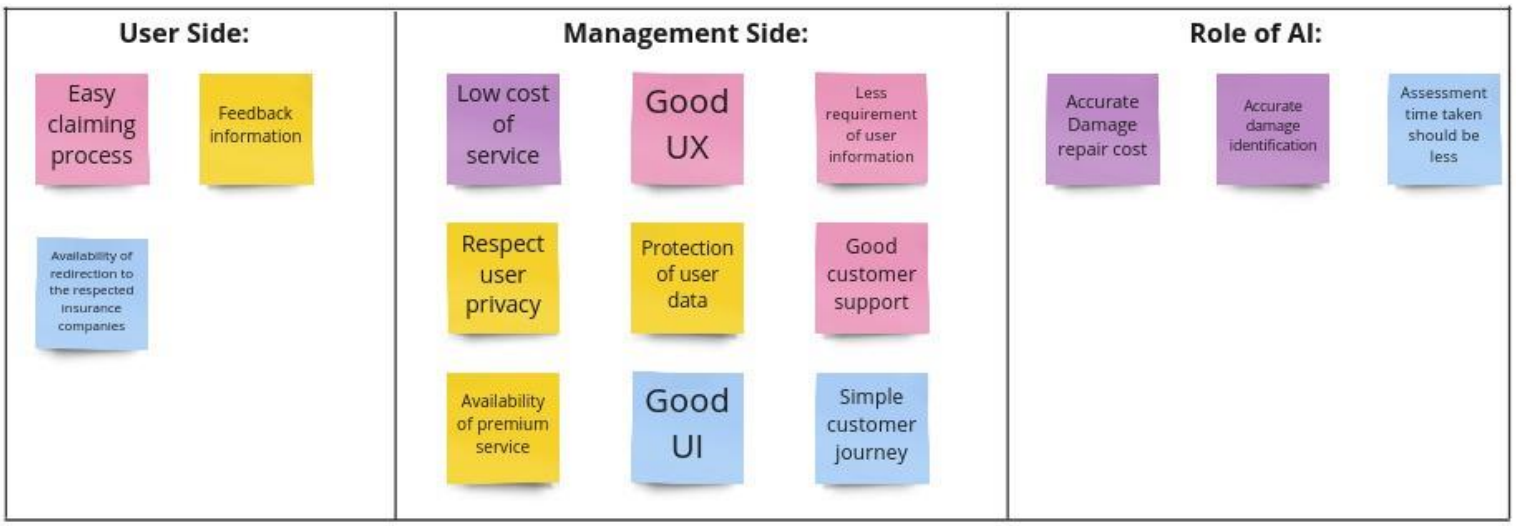
Problem Solution:

A site which helps in getting insurance amount for damaged vehicles

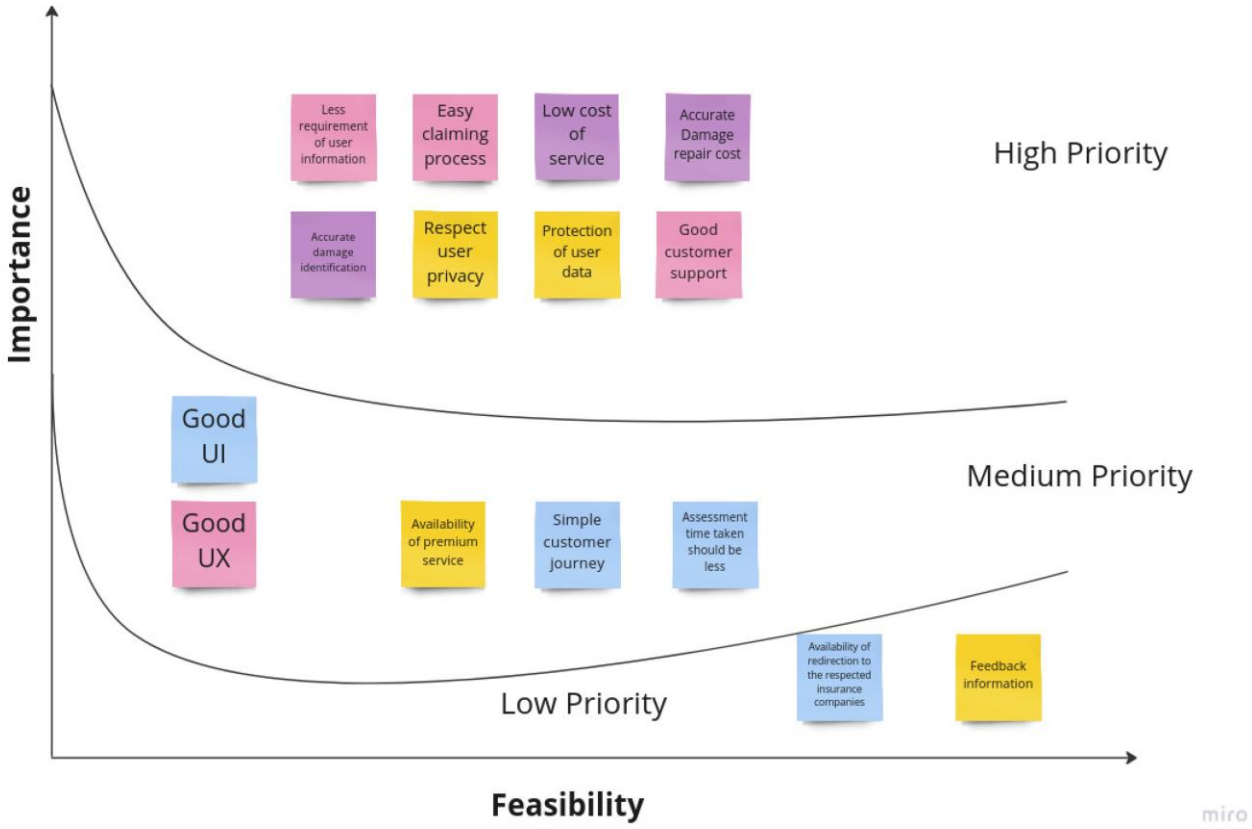
Brainstorming Session:



Idea Grouping:



Idea Prioritization :



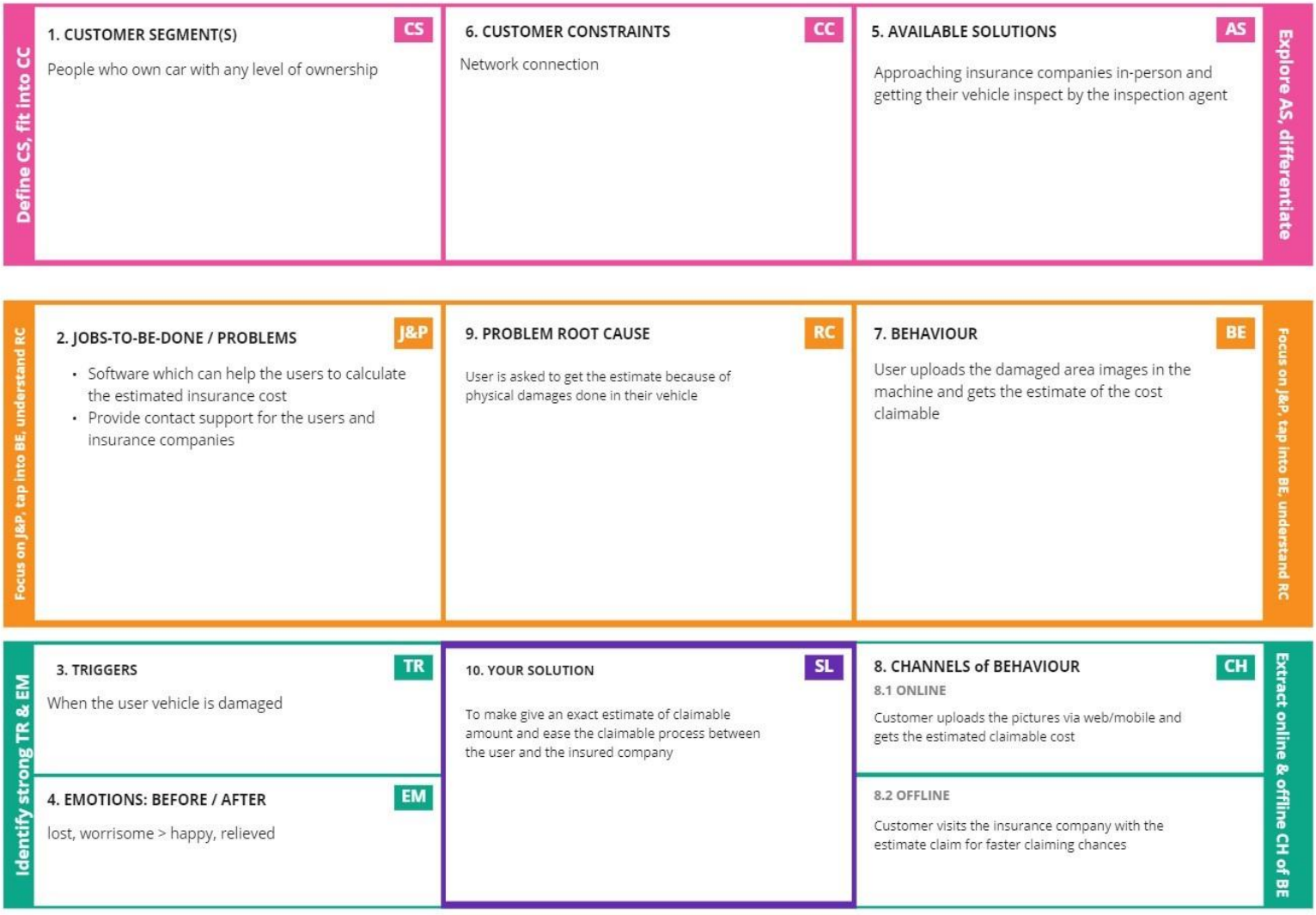
3.3. Proposed Solution

The proposed solution for the problems faced by the customers during the estimation of the cost of the damage is given in the below table

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To create a site which helps the user to get an estimate of claimable insurance amount for the damaged vehicle and measure the level of damage
2.	Idea / Solution description	To create a site using python and ibm cloud tools which enables the user to upload damaged vehicle pictures and get the estimated insurance amount claimable
3.	Novelty / Uniqueness	Less user information is obtained while estimate documentation and site redirection to the respective insured company is also provided
4.	Social Impact / Customer Satisfaction	User gets the claimable amount and the claim amount worries are solved
5.	Business Model (Revenue Model)	Customer service is provided for users during the whole insurance process as a part of Premium Subscription
6.	Scalability of the Solution	Insurance service can be provided in the long run with better return benefits than various insurance companies

3.3. Problem Solution Fit

The solution fit for the problem that discusses various customer aspects is represented using the below image which was generated using miro



4. REQUIREMENT ANALSIS

4.1. Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	Registration through website using Gmail
FR-2	**User Confirmation**	Confirmation via Email
FR-3	**User Details**	Entering the vehicle registration and license details on the website
FR-4	**Damage Detector**	User uploads the image of the damaged area of the vehicle on the website and waits for the results
FR-5	**User Solution**	The level and the estimation of the damaged vehicle can be determined explicitly
FR-6	**User Advantage**	Using the results of the website the user can approach the insurance company to get the claimable insurance amount and repair the damage

4.2. Non-Functional Requirements

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Estimate the level and the cost of the damage that has been made on the vehicle
NFR-2	**Security**	Safe and secure protection of the details provided by the user
NFR-3	**Reliability**	The estimation of the model is mostly precise
NFR-4	**Performance**	The processing of the image takes a little bit of a time but it does not make the user displease
NFR-5	**Availability**	The website is 24/7 available so that it can be accessed by the user any time
NFR-6	**Scalability**	Can get the proper estimation of the damage and can approach the insurance companies for better service

5. PROJECT DESIGN

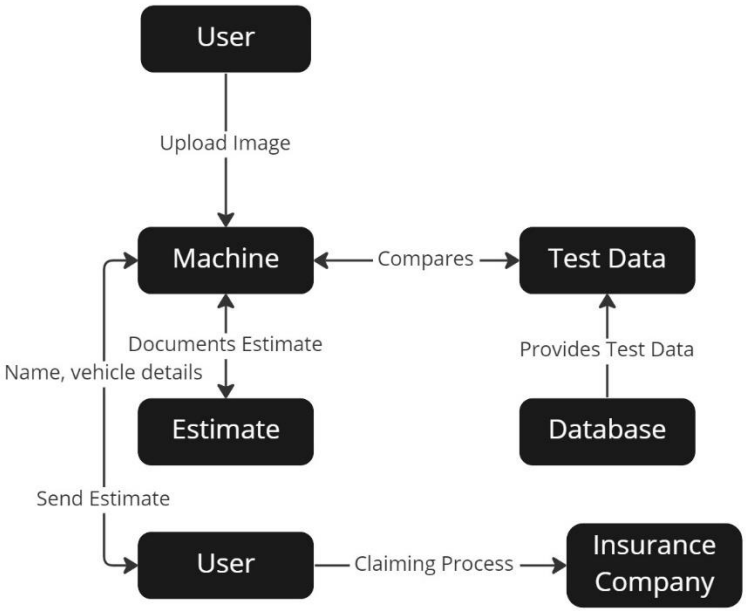
5.1. Dataflow Diagrams

Data Flow Diagram:

The classic visual representation of how information moves through a system is a data flow diagram (DFD). A tidy and understandable DFD can graphically represent the appropriate quantity of the system demand. It can be done manually, automatically, or both. It demonstrates how information enters and exits the system, what modifies the data, and where information is kept. A DFD's goal is to outline the boundaries and scope of a system as a whole. The below flow depicts how the user requests his/her request.

Flow:

1. The user first uploads the picture of the damaged area in the car to the website.
2. The machine which is trained using the dataset analyses the photo that is uploaded.
3. Then it gives the estimation and the severity of the damage on the vehicle.
4. Using the estimation given by the AI, the customer will be able to approach the insurance company and claim the insurance amount for the restoration of the vehicle.



5.2. Solution and Technical Architecture

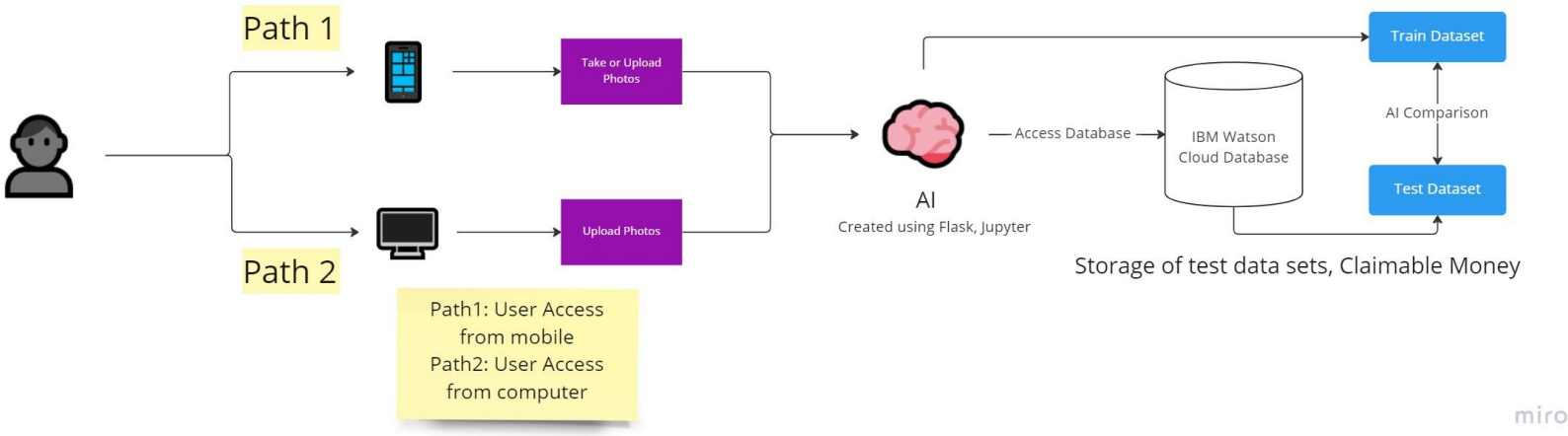
A complicated process with numerous sub-processes, solution architecture connects business issues with technological solutions. Its objectives are to:

• Identify the best technological solution to address current business issues.

• Explain to project stakeholders the structure, traits, behavior, and other features of the software.

• Specify the features, stages of development, and requirements for the solution.

• Offer guidelines for how the solution is created, managed, and delivered.



5.3. User Stories

The user stories regarding the working of the website that is developed is represented in the following table

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web User)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can enter the details of my vehicle In the website.	I can register & access the details of the vehicle	High	Sprint-1
		USN-4	As a user, I can enter the license and registration of my vehicle for the insurance process	I can register the registration and license details on the website with proper proofs	Medium	Sprint-2
	Upload images	USN-5	As a user, I can upload the pictures of the damage that has been caused in my vehicle through the website	The pictures of the damaged vehicle that is yours can only be uploaded	High	Sprint-2
	Dataset Collection	USN-6	The website is trained using a training dataset so that it can precisely give the estimation of the damage	The damage in the vehicle must be captured clearly in the camera	High	Sprint-2
		USN-7	The level and the estimation of the damage is given by the website using the trained algorithms and dataset	Clear level and estimation of damage is given	Medium	Sprint-3
	Output	USN-8	As a customer with the results obtained from the website, they can approach the claimable amount from the respected insurance companies	Insurance amount and the damage can be rectified by the companies	High	Sprint-4
	Experience	USN-9	As a customer it provides full satisfaction of the website	Provide more feedback and better results in future	High	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

6.1. Sprint Planning and Estimation

The sprint planning for each sprint is given below

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Registration	USN-1	As an owner of a particular vehicle, I can log into the website by entering email & password.	2	Medium	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-2	User Confirmation	USN-2	As an owner of a particular Vehicle, I will receive confirmation email once I have registered for the website.	1	Medium	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-2	Login	USN-3	As an owner of a particular vehicle, I can log into the website by entering email & password.	2	Low	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-1	Data Collection	USN-1	Download the dataset used in intelligent vehicle damage assessment & cost estimator for insurance companies.	2	High	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-1	Image Pre Processing	USN-1	Improve the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, etc.	2	High	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand

Sprint-1	Model Building	USN-1	Define the model architecture and adding CNN layer and testing ,saving the model.	2	High	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-3	Cloud DB	USN-1	Below are steps that need to follow for creating and using cloudant service. • Register & login to IBM cloud • Create service instance • Creating service credentials • Launch cloudant DB • Create database	2	High	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-4	Application Building	USN-1	Building a web application that is integrated into the model we built. A UI is provided to the user where he has uploaded the image. Based on the saved model, the uploaded image will be analyzed and prediction is showcased on the UI.	2	High	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand
Sprint-4	Train The Model On IBM	USN-1	Build Deep learning model and computer vision Using the IBM cloud.	2	High	Sanjay V Sathish Kumar G Sanjay Preeth D Dhivyesh Anand

6.2. Sprint Delivery Schedule

The delivery of the sprint plan is given along with its date

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	5 Days	28Oct 2022	03Oct 2022	20	04Oct 2022
Sprint-2	20	6 Days	02Nov 2022	09Nov2022	20	09 Nov2022
Sprint-3	20	8 Days	09Nov2022	17Nov2022	20	17Nov2022
Sprint-4	20	8 Days	10 Nov 2022	19Nov2022	20	19 Nov 2022

7. CODING AND SOLUTIONING

Forgotpassword.html

```
<html>
  <head>
    <title>Login</title>
    <style type="text/css">
    body{
        background-image: linear-gradient(90deg, #094f79 5%, #00ff84 100%);
          }
      #topmenu {
        width: 100%;
        background-color: 312D2D;
        height: 50px;
      }
      #hedder {
        color: white;
        font-size: large;
        padding-top: 13px;
        padding-left: 40px;
      }
      #home {
        float: right;
        padding-top: 13px;
        padding-right: 50px;
```

```
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#box {
  height: 300px;
  width: 500px;
  background-color: antiquewhite;
  margin: 10px;
  border-color: black;
  border-width: 25px;
}
```

```
    div.background {
      border: 2px solid gray;
      height: 300px;
      width: 500px;
      margin: auto;
      margin-top: 7%;
    }
    #loginlogo {
      text-align: center;
      margin-top: 20px;
    }
    #textcontent {
      margin-top: 10px;
      margin-left: 25px;
      margin-top: 20px;
    }
  </style>
  <script>
 window.watsonAssistantChatOptions = {
  integrationID: "1d4165da-e3a9-481a-9c00-f93fde429ffe", // The ID of this integration.
  region: "us-south", // The region your integration is hosted in.
  serviceInstanceID: "408f94ed-164a-4206-ab8c-91955c05e343", // The ID of your service instance.
  onLoad: function(instance) { instance.render(); }
 };
```

```
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
  </head>
  <body onload="flashMessage()">
    <div id="topmenu">
      <div id="register">
        <a href="/register" style="color: white;text-decoration: none;">Register</a>
      </div>
      <div id="login">
        <a href="/login" style="color: white;text-decoration: none;">Logout</a>
      </div>
      <div id="home">
        <a href="/" style="color: white;text-decoration: none;">Home</a>
      </div>
      <div id="hedder">Login Page</div>
    <!--</div>
    {% with messages = get_flashed_messages() %}
  {% if messages %}
    <ul class=flashes>
```

```
    {% for message in messages %}
      <p><strong>Error:</strong> {{ message }}
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}-->

    <div class="background">
      <div id="loginlogo">
        <img src="static/3275434.jpg" width="150" height="150"></img>
        <lord-icon
        src="static/3275434.jpg"
        trigger="hover"
        style="width:100px;height:100px">
        </lord-icon>
        <!-- <img
          src= "/static/images/login icon.png"
          alt="login logo"
          style="width: 100px; height: 100px; border-radius: 50%"
        /> -->
      </div>
      <div id="textcontent">

        <form action="login" method="POST">
          <script>
```

```
  function flashMessage(){
    if("{{flash_message}}" == "True"){
      alert("invalid credentials")
    }
  }
</script>
<input
  type="text"
  name="email"
  id="email"
  placeholder="Enter Registered Email ID"
  style=" border-radius:15px; width: 440px; height: 35px; margin-bottom: 15px"
/>
<input
  type="submit"
  name="submit"
  value="Submit"
  style="
  border-radius:15px;
    width: 440px;
    height: 35px;
    text-align: center;
    background-color: black;
    color: white;
  "
```

```
        />
       </form>
      </div>
    </div>
    </div>
  </body>
</html>
```

IBM WATSON:

```
  window.watsonAssistantChatOptions = {
    integrationID: "1d4165da-e3a9-481a-9c00-f93fde429ffe", // The ID of this integration.
    region: "us-south", // The region your integration is hosted in.
    serviceInstanceID: "408f94ed-164a-4206-ab8c-91955c05e343", // The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
```

Additionally we have also added ibm Watson to guide the user regarding any queries

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.

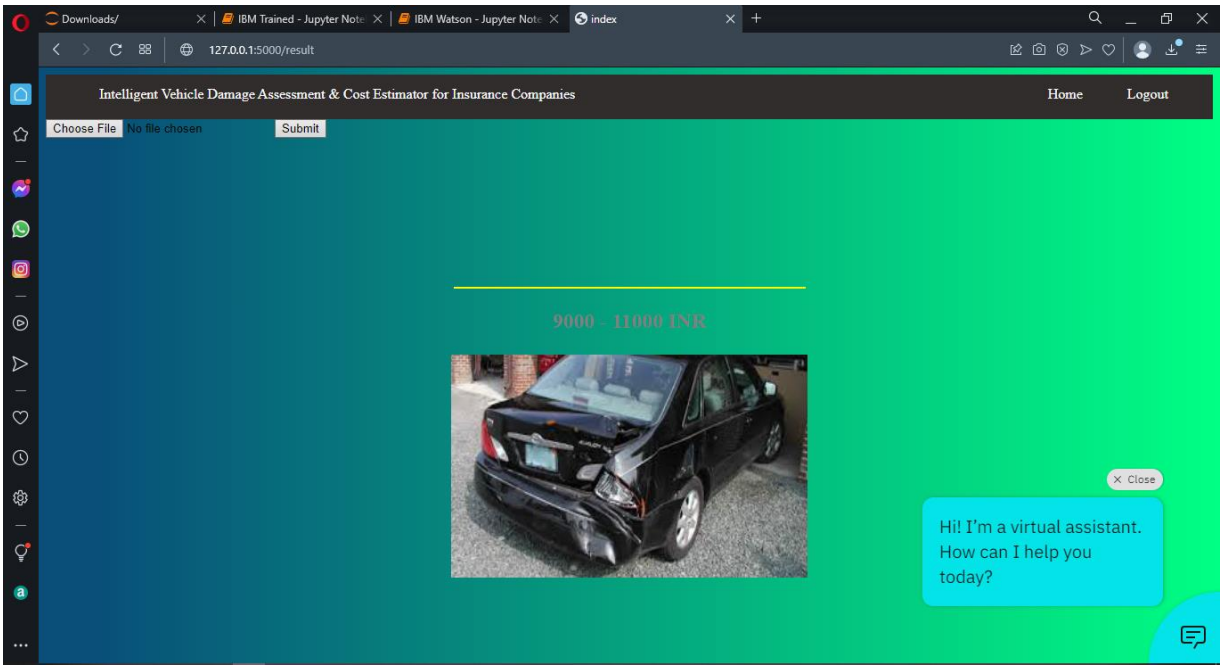
Reusable: Can be reused if necessary

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

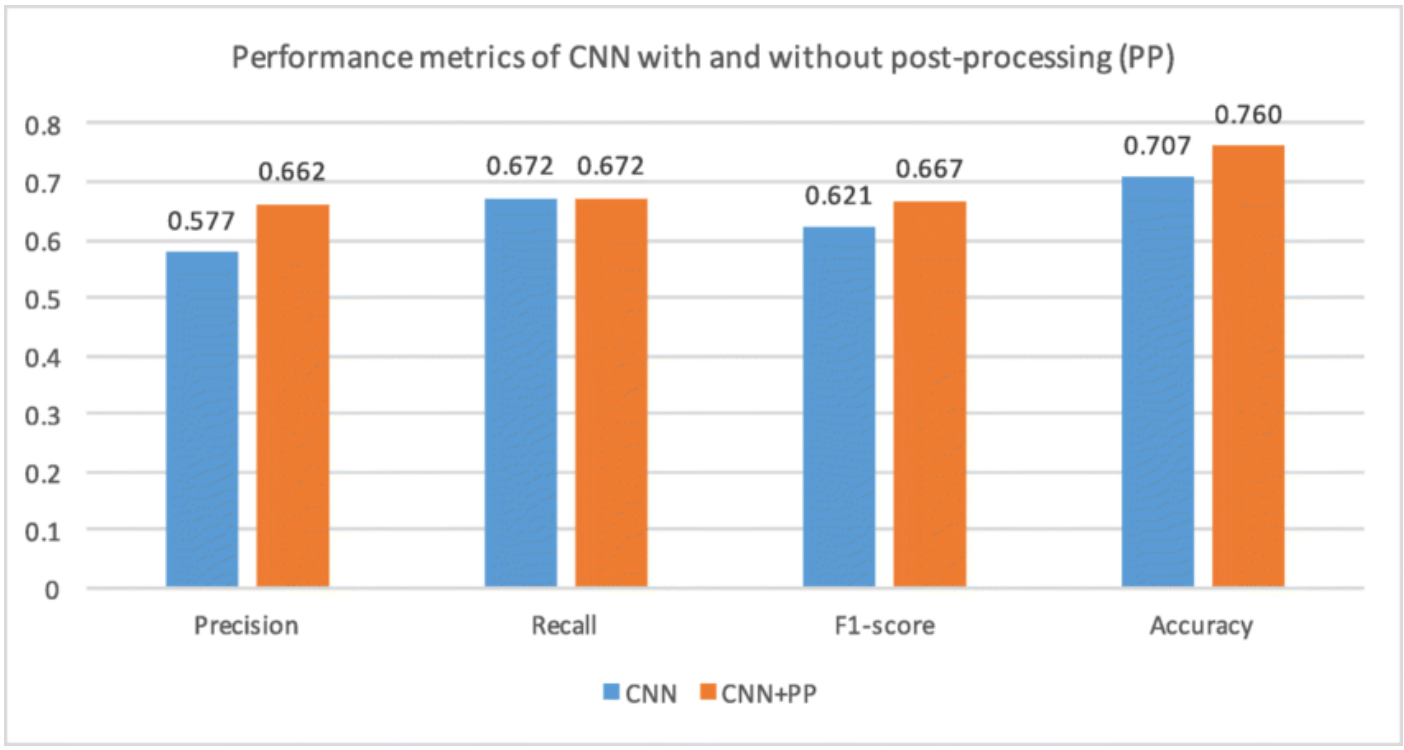
Users, clients, or other authorized organizations conduct this type of testing to determine the specifications and operational practices of an application or piece of software. Acceptance testing is the most important testing phase since it establishes whether or not the customer will accept the application or software. It could involve the user interface, functionality, usability, and usefulness of the application. It is also known as operational acceptance testing, user acceptability testing, and end-user testing (UAT).

9. RESULTS



9.1 Performance Testing

Performance metrics are defined as figures and data representative of an organization's actions, abilities, and overall quality.



10. ADVANTAGES AND DISADVANTAGES

10.1. Advantages

- Provides a detailed cost for the damage on the vehicle.
- Digitalized claim process is easy to understand
- Give the accurate result of the damaged area on the vehicle
- Helps the insurance company to analyze the damaged vehicle and also payment process.
- User can get respected knowledge regarding the vehicle damage insurance

10.2. Disadvantages

- Needs more development and features
- It will take more time to claim the insurance in manual process
- Requires internet connection to process of registration
- The insurance company should allow this kind of approaches in the industry

11. CONCLUSION

11.1. Conclusion

We proposed a solution using ML models to predict claim occurrence in the next year and to adjust the insurance prices fairly to the client's ability, and used relevant personal information. The pre-trained models were put to the test using transfer learning and fine-tuning with specific regularization strategies. We may confidently infer from the aforementioned models that the image classification model, VGG16 models, and dense net models work best for determining if an automobile has been damaged or not, the classification of the damage to the car, and the severity of the damage.

Additionally, we may attempt to anticipate the cost of repairing for the damaged car part if we have a sufficient high-quality dataset with adequate characteristics and labels, which would assist the auto-insurance business in coming up with better and more affordable alternatives.

12. FUTURE SCOPE

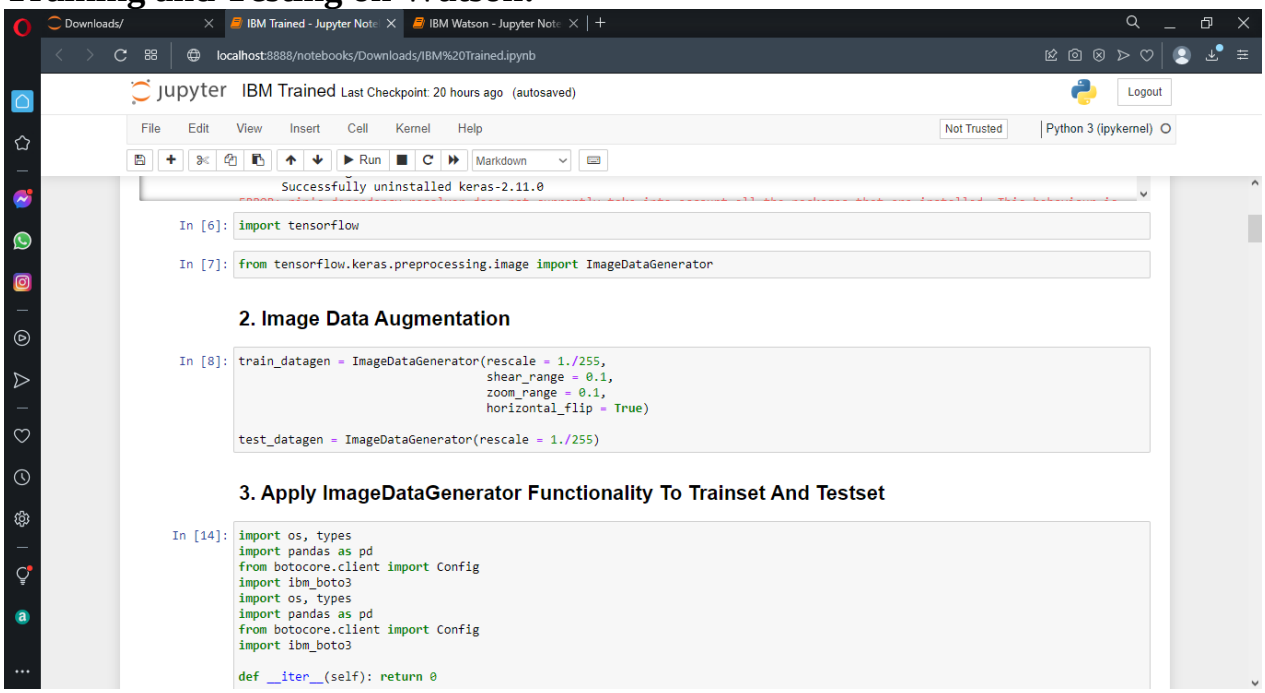
12.1. Future scope

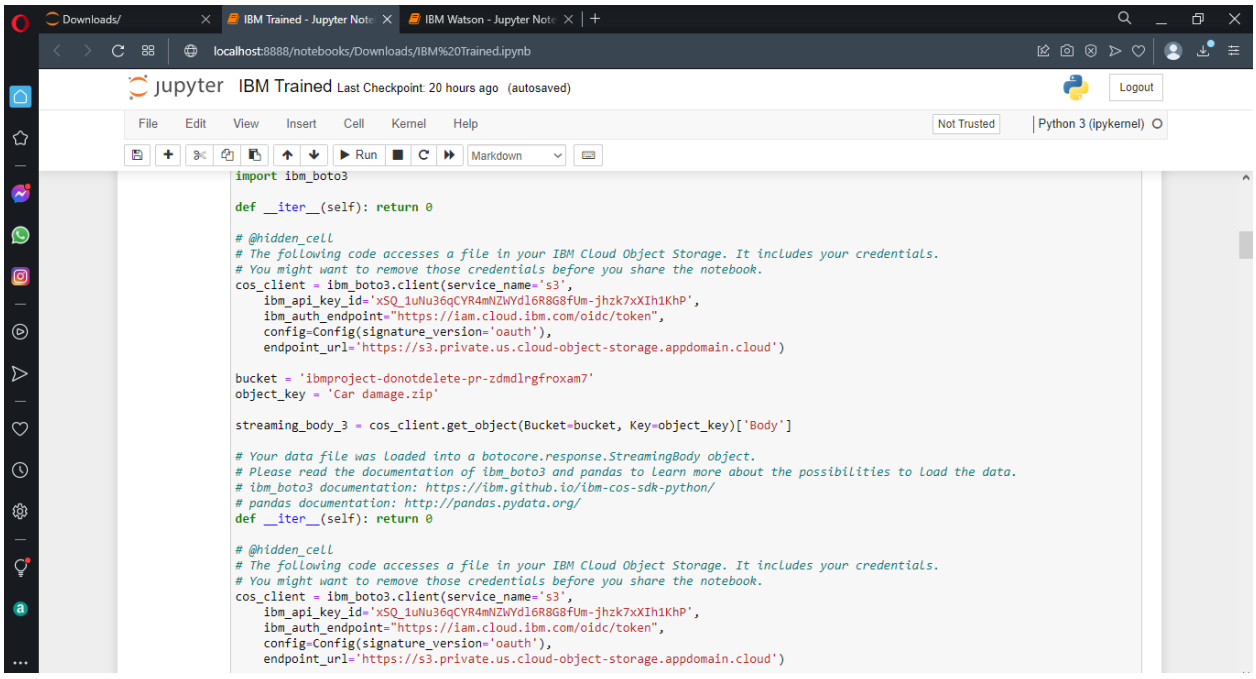
It would be useful to see comparative studies for other ML and deep learning models using this dataset; it also would be worth performing this analysis with another insurance branch to conclude whether random forests still have the best predictive output or not. It would be a benefit for both the insurance company and the customer as it also reduces any manual work and time.

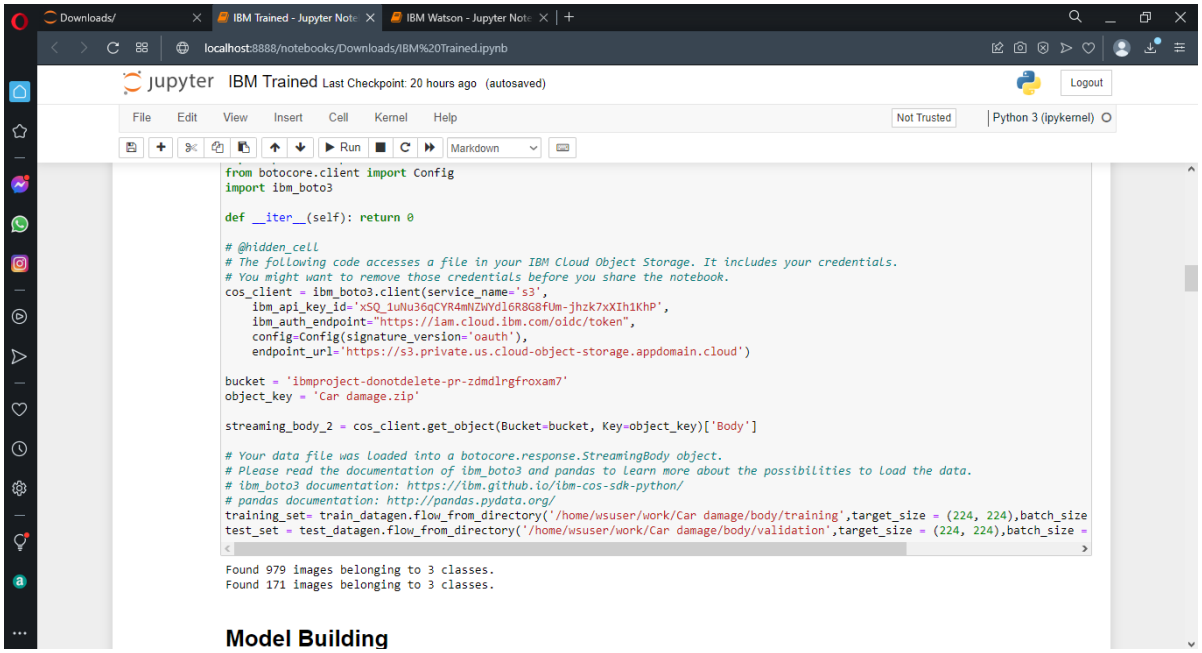
13. APPENDIX

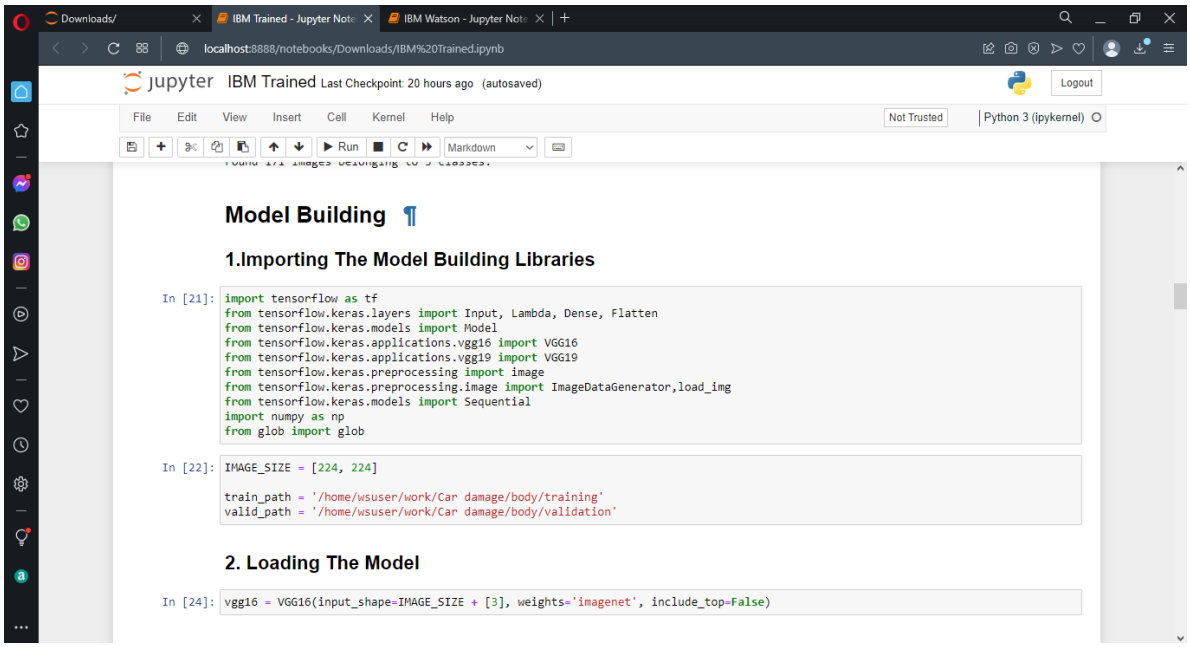
Source Code

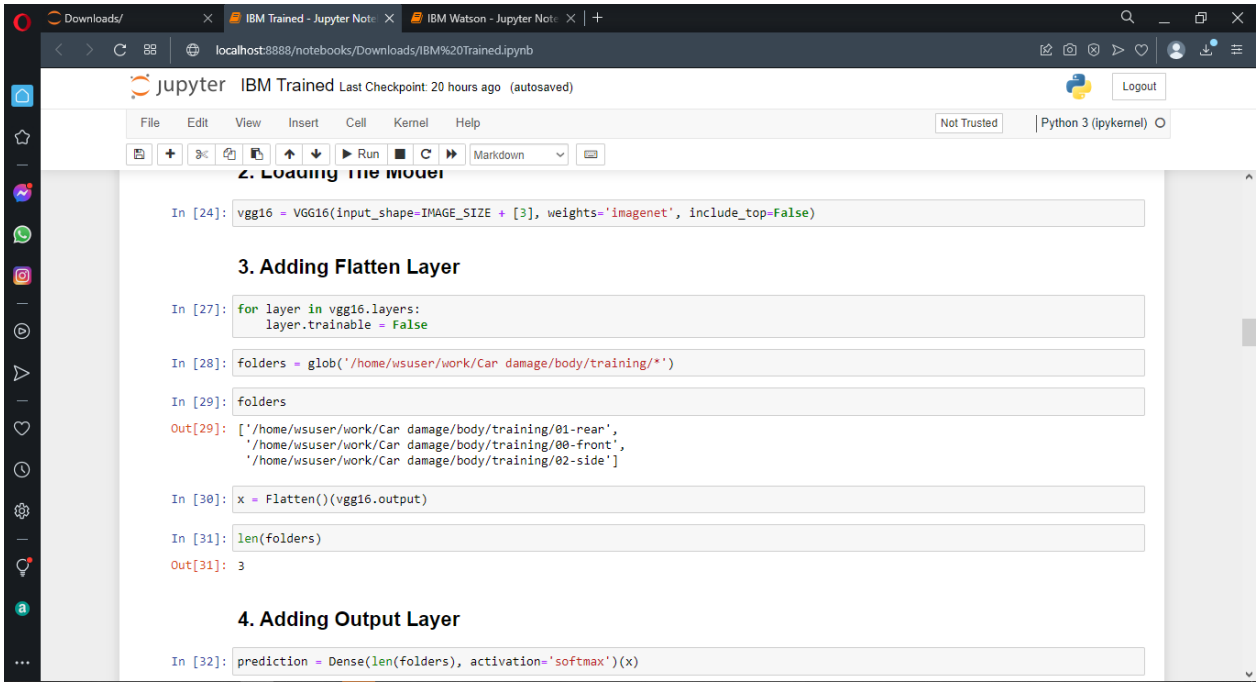
Training and Testing on Watson:

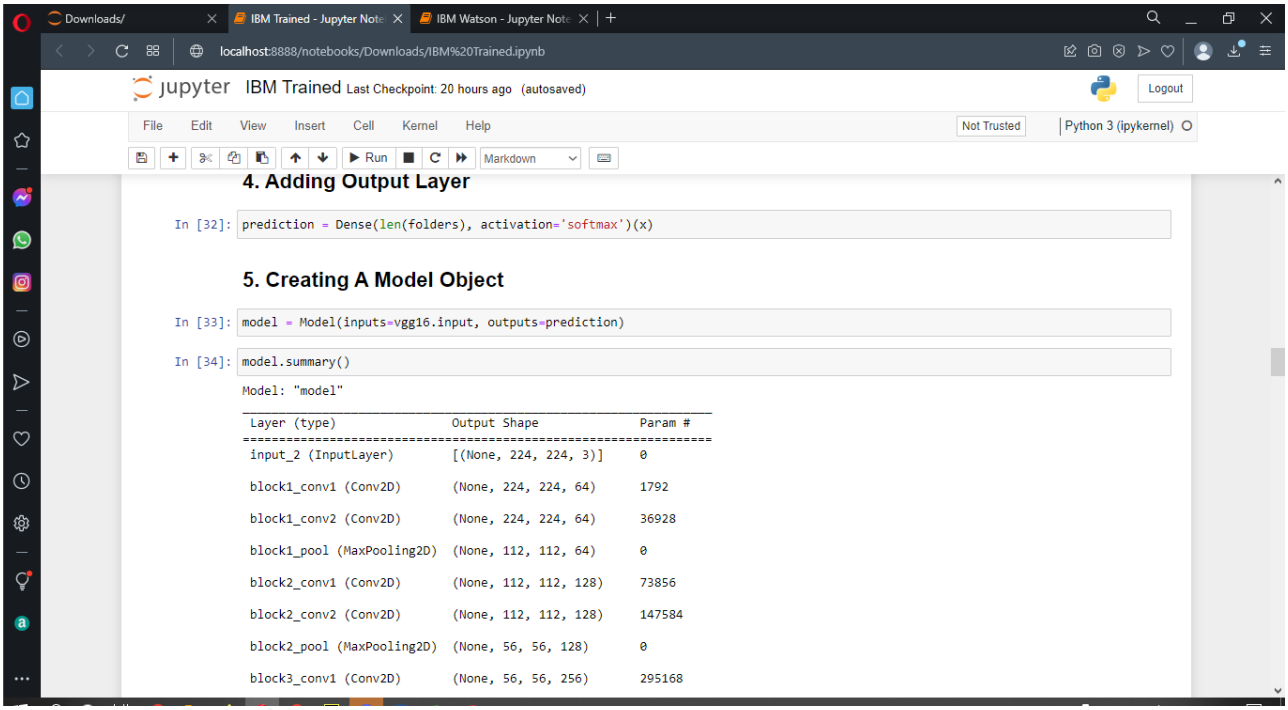


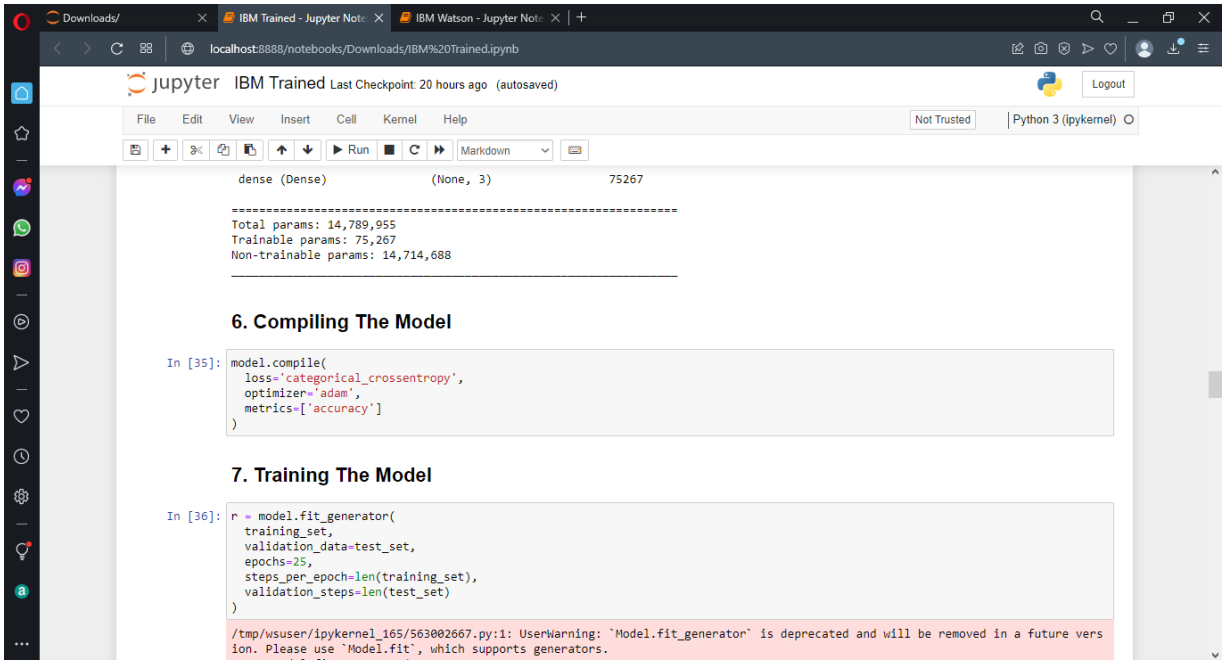


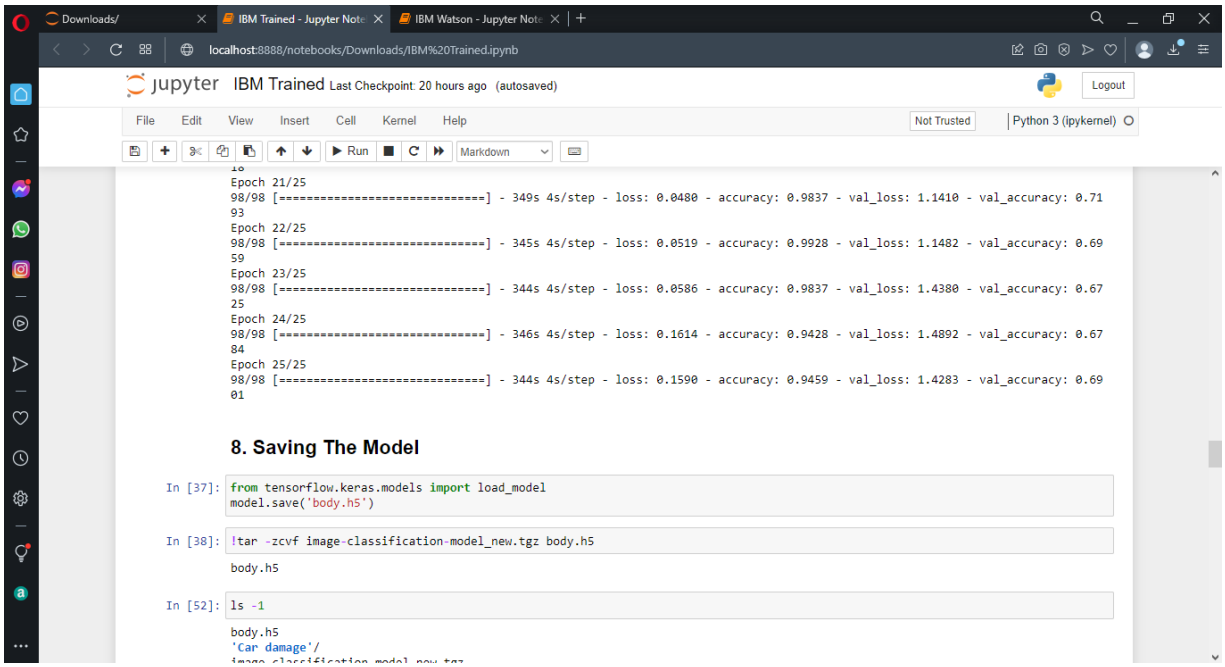

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
from botocore.client import Config
import ibm_boto3
def __iter__(self): return 0
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
ibm_api_key_id='xSQ_1uNu36qCYR4mNZWYdl6R8G8fUm-jhzk7xXIh1KhP',
ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
config=Config(signature_version='oauth'),
endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
bucket = 'ibmproject-donotdelete-pr-zdmdlrgfroxam7'
object_key = 'Car damage.zip'
streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
training_set= train_datagen.flow_from_directory('/home/wsuser/work/Car damage/body/training',target_size = (224, 224),batch_size
test_set = test_datagen.flow_from_directory('/home/wsuser/work/Car damage/body/validation',target_size = (224, 224),batch_size =
Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.
Model Building

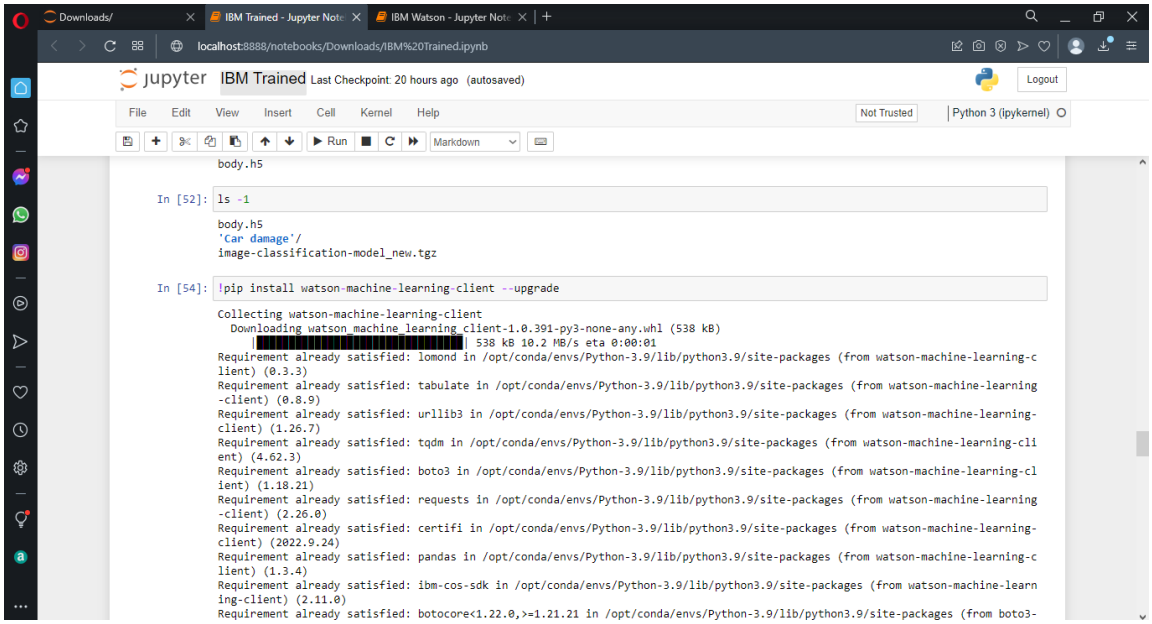

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Model Building ¶
1.Importing The Model Building Libraries
In [21]:
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
In [22]:
IMAGE_SIZE = [224, 224]
train_path = '/home/wsuser/work/Car damage/body/training'
valid_path = '/home/wsuser/work/Car damage/body/validation'
2. Loading The Model
In [24]:
vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

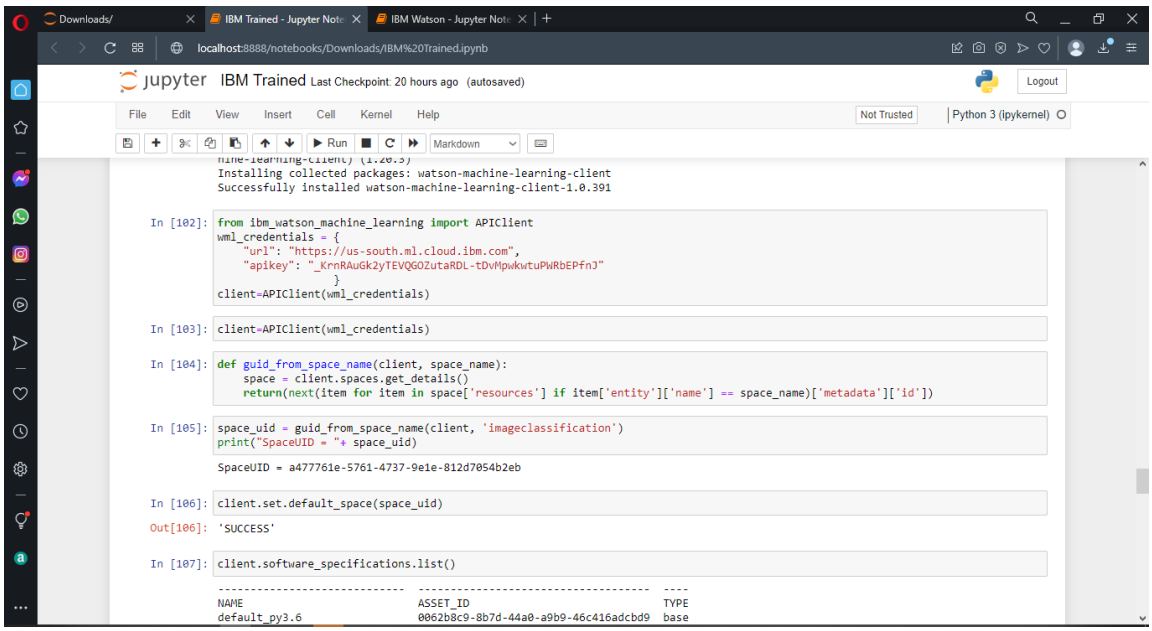

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
2. Loading The Model
In [24]: vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
3. Adding Flatten Layer
In [27]: for layer in vgg16.layers:
layer.trainable = False
In [28]: folders = glob('/home/wsuser/work/Car damage/body/training/*')
In [29]: folders
Out[29]: ['/home/wsuser/work/Car damage/body/training/01-rear',
'/home/wsuser/work/Car damage/body/training/00-front',
'/home/wsuser/work/Car damage/body/training/02-side']
In [30]: x = Flatten()(vgg16.output)
In [31]: len(folders)
Out[31]: 3
4. Adding Output Layer
In [32]: prediction = Dense(len(folders), activation='softmax')(x)

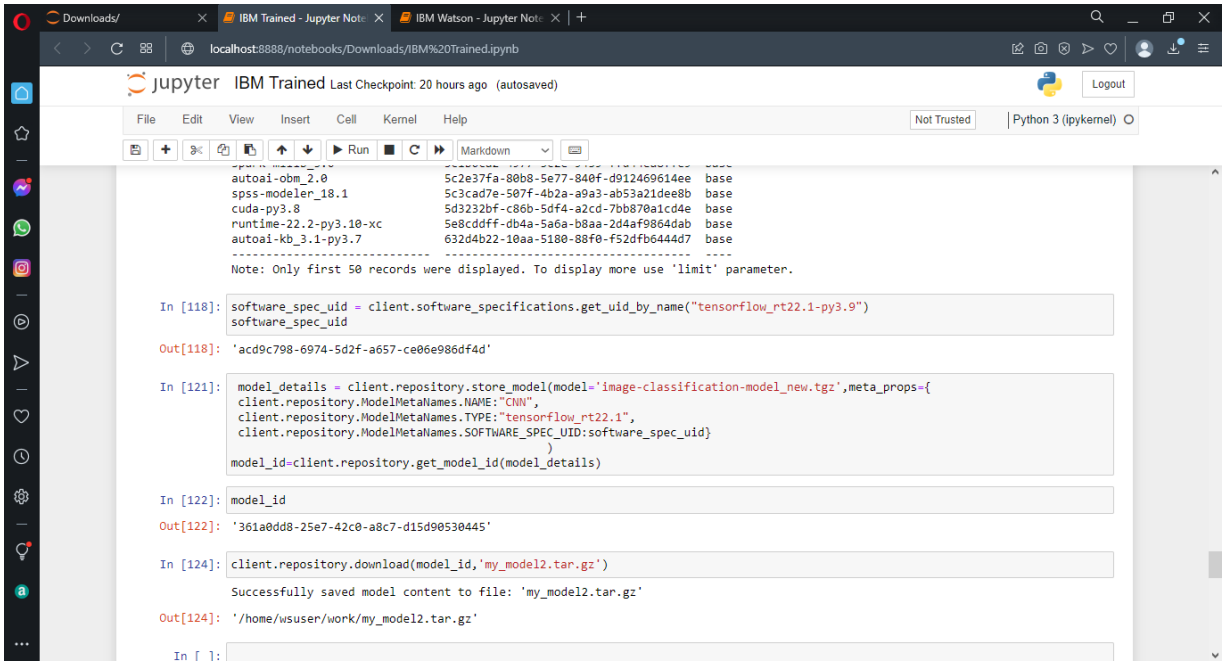

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
4. Adding Output Layer
In [32]: prediction = Dense(len(folders), activation='softmax')(x)
5. Creating A Model Object
In [33]: model = Model(inputs=vgg16.input, outputs=prediction)
In [34]: model.summary()
Model: "model"
Layer (type) Output Shape Param #
input_2 (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

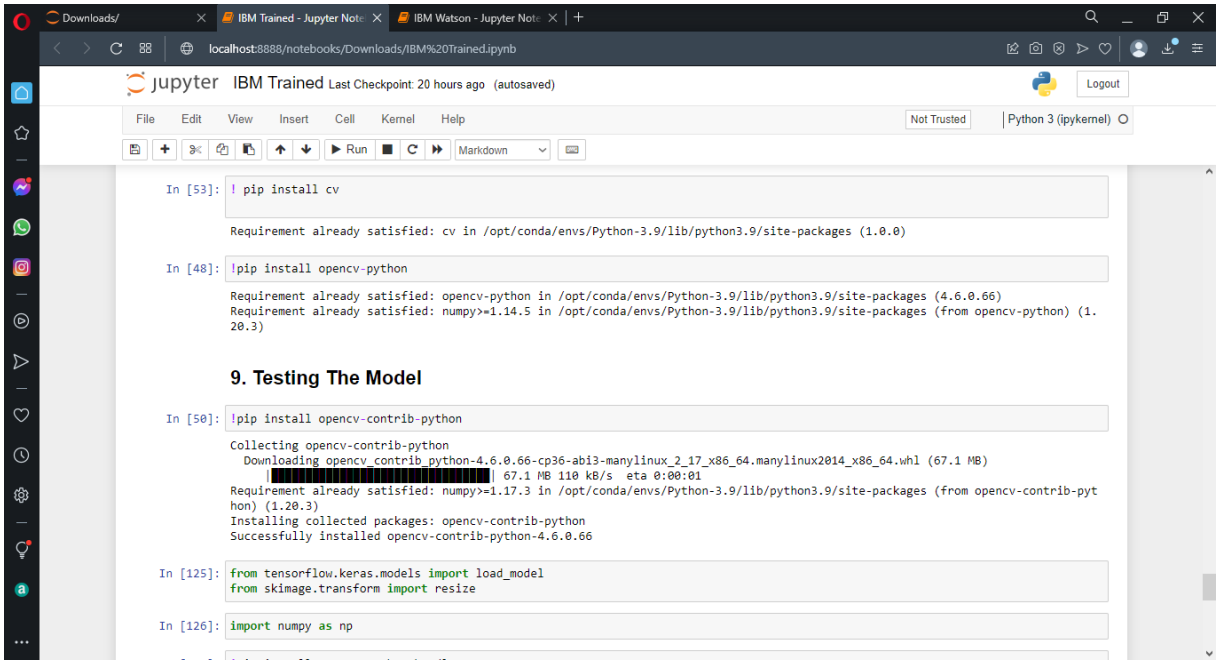

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
dense (Dense) (None, 3) 75267
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
6. Compiling The Model
In [35]:
model.compile(
loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy']
)
7. Training The Model
In [36]:
r = model.fit_generator(
training_set,
validation_data=test_set,
epochs=25,
steps_per_epoch=len(training_set),
validation_steps=len(test_set)
)
/tmp/wsuser/ipykernel_165/563002667.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future vers
ion. Please use `Model.fit`, which supports generators.


Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Epoch 21/25
98/98 [==========] - 349s 4s/step - loss: 0.0480 - accuracy: 0.9837 - val_loss: 1.1410 - val_accuracy: 0.71
93
Epoch 22/25
98/98 [==========] - 345s 4s/step - loss: 0.0519 - accuracy: 0.9928 - val_loss: 1.1482 - val_accuracy: 0.69
59
Epoch 23/25
98/98 [==========] - 344s 4s/step - loss: 0.0586 - accuracy: 0.9837 - val_loss: 1.4380 - val_accuracy: 0.67
25
Epoch 24/25
98/98 [==========] - 346s 4s/step - loss: 0.1614 - accuracy: 0.9428 - val_loss: 1.4892 - val_accuracy: 0.67
84
Epoch 25/25
98/98 [==========] - 344s 4s/step - loss: 0.1590 - accuracy: 0.9459 - val_loss: 1.4283 - val_accuracy: 0.69
01
8. Saving The Model
In [37]:
from tensorflow.keras.models import load_model
model.save('body.h5')
In [38]:
!tar -zcvf image-classification-model_new.tgz body.h5
body.h5
In [52]:
ls -1
body.h5
'Car damage'/


Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
body.h5
In [52]: ls -1
body.h5
'Car damage'/
image-classification-model_new.tgz
In [54]: !pip install watson-machine-learning-client --upgrade
Collecting watson-machine-learning-client
Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
538 kB 10.2 MB/s eta 0:00:01
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.9.24)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3-


Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
In [102]: from ibm_watson_machine_learning import APIClient
wml_credentials = {
"url": "https://us-south.ml.cloud.ibm.com",
"apikey": "_KrnRAuGk2yTEVQGOZutaRDL-tDvMpwkwtuPWRbEPfnJ"
}
client=APIClient(wml_credentials)
In [103]: client=APIClient(wml_credentials)
In [104]: def guid_from_space_name(client, space_name):
space = client.spaces.get_details()
return(next(item for item in space['resources'] if item['entity']['name'] == space_name)['metadata']['id'])
In [105]: space_uid = guid_from_space_name(client, 'imageclassification')
print("SpaceUID = "+ space_uid)
SpaceUID = a477761e-5761-4737-9e1e-812d7054b2eb
In [106]: client.set.default_space(space_uid)
Out[106]: 'SUCCESS'
In [107]: client.software_specifications.list()
NAME ASSET_ID TYPE
default_py3.6 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base


Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
autoai-obm_2.0 5c2e37fa-80b8-5e77-840f-d912469614ee base
spss-modeler_18.1 5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b base
cuda-py3.8 5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e base
runtime-22.2-py3.10-xc 5e8cddff-db4a-5a6a-b8aa-2d4af9864dab base
autoai-kb_3.1-py3.7 632d4b22-10aa-5180-88f0-f52dfb6444d7 base
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
In [118]:
software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
Out[118]:
'acd9c798-6974-5d2f-a657-ce06e986df4d'
In [121]:
model_details = client.repository.store_model(model='image-classification-model_new.tgz',meta_props={
client.repository.ModelMetaNames.NAME:"CNN",
client.repository.ModelMetaNames.TYPE:"tensorflow_rt22.1",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}
)
model_id=client.repository.get_model_id(model_details)
In [122]:
model_id
Out[122]:
'361a0dd8-25e7-42c0-a8c7-d15d90530445'
In [124]:
client.repository.download(model_id,'my_model2.tar.gz')
Successfully saved model content to file: 'my_model2.tar.gz'
Out[124]:
'/home/wsuser/work/my_model2.tar.gz'
In [ ]:


Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb
jupyter IBM Trained Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
In [53]:
! pip install cv
Requirement already satisfied: cv in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.0)
In [48]:
!pip install opencv-python
Requirement already satisfied: opencv-python in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from opencv-python) (1.20.3)
9. Testing The Model
In [50]:
!pip install opencv-contrib-python
Collecting opencv-contrib-python
Downloading opencv_contrib_python-4.6.0.66-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (67.1 MB)
67.1 MB 110 kB/s eta 0:00:01
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from opencv-contrib-python) (1.20.3)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-4.6.0.66
In [125]:
from tensorflow.keras.models import load_model
from skimage.transform import resize
In [126]:
import numpy as np

Downloads/ | IBM Trained - Jupyter Note | IBM Watson - Jupyter Note

localhost:8888/notebooks/Downloads/IBM%20Trained.ipynb

jupyter **IBM Trained** Last Checkpoint: 20 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help | Not Trusted | Python 3 (ipykernel)

Run | Markdown

In [128]:
```
!pip install opencv-python-headless
```

```
Collecting opencv-python-headless
  Downloading opencv_python_headless-4.6.0.66-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (48.3 MB)
     |████████████████████████████████| 48.3 MB 16.4 MB/s eta 0:00:01
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from opencv-python-head
less) (1.20.3)
Installing collected packages: opencv-python-headless
Successfully installed opencv-python-headless-4.6.0.66
```

In [129]:
```
import cv2
```

In [130]:
```
model = load_model('body.h5')
```

In [131]:
```
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
```

In [132]:
```
import numpy as np
```

In [133]:
```
from PIL import Image
```

In [137]:
```
Image.open("/home/wsuser/work/Car damage/body/validation/01-rear/0004.JPEG")
```

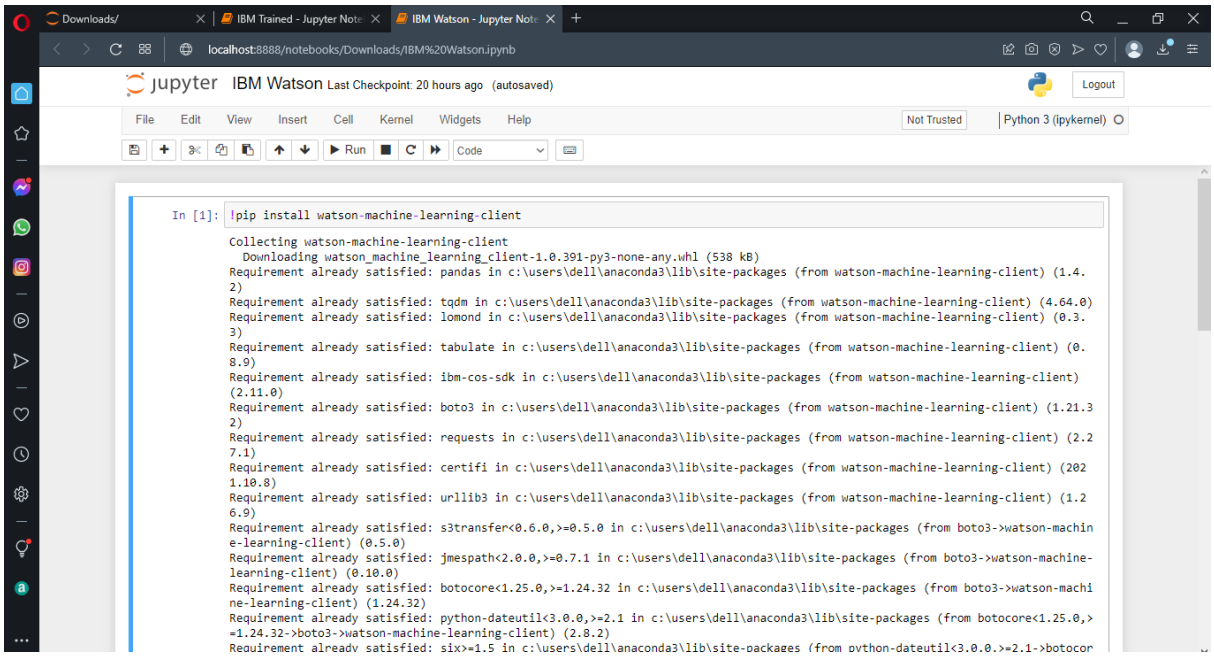
Out[137]:

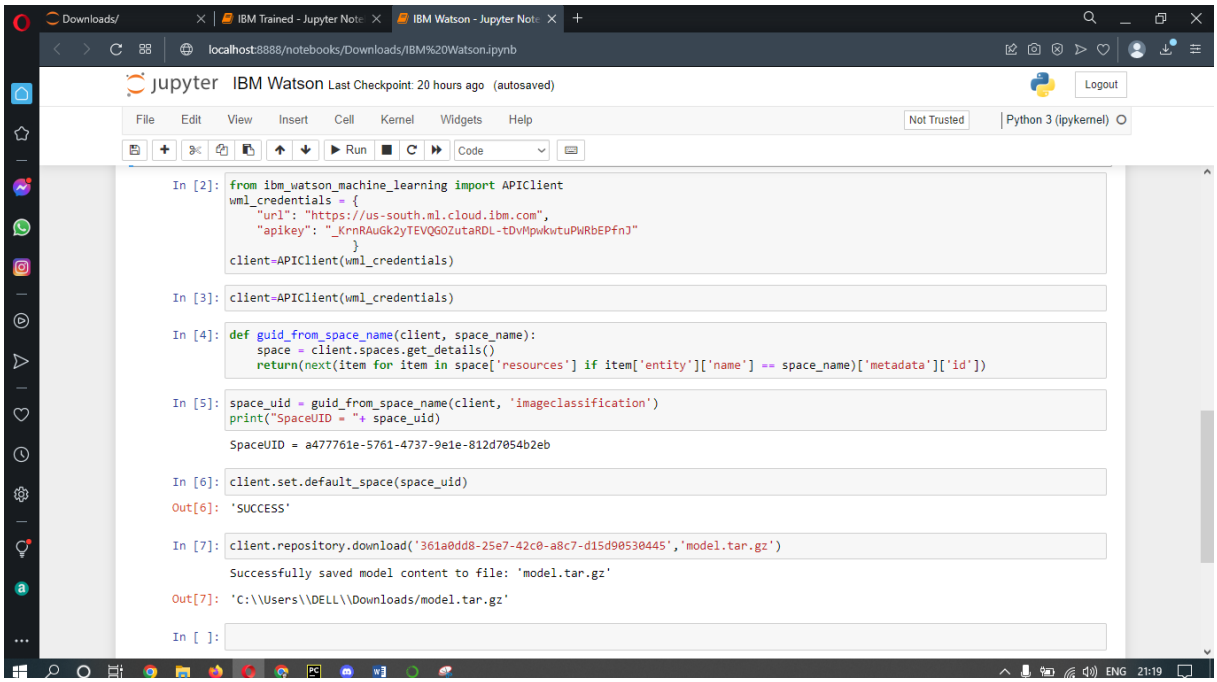
In [138]:
```
data = "/home/wsuser/work/Car damage/body/validation/01-rear/0004.JPEG"
image = cv2.imread(data)
print(detect(image))
```

```
rear
```

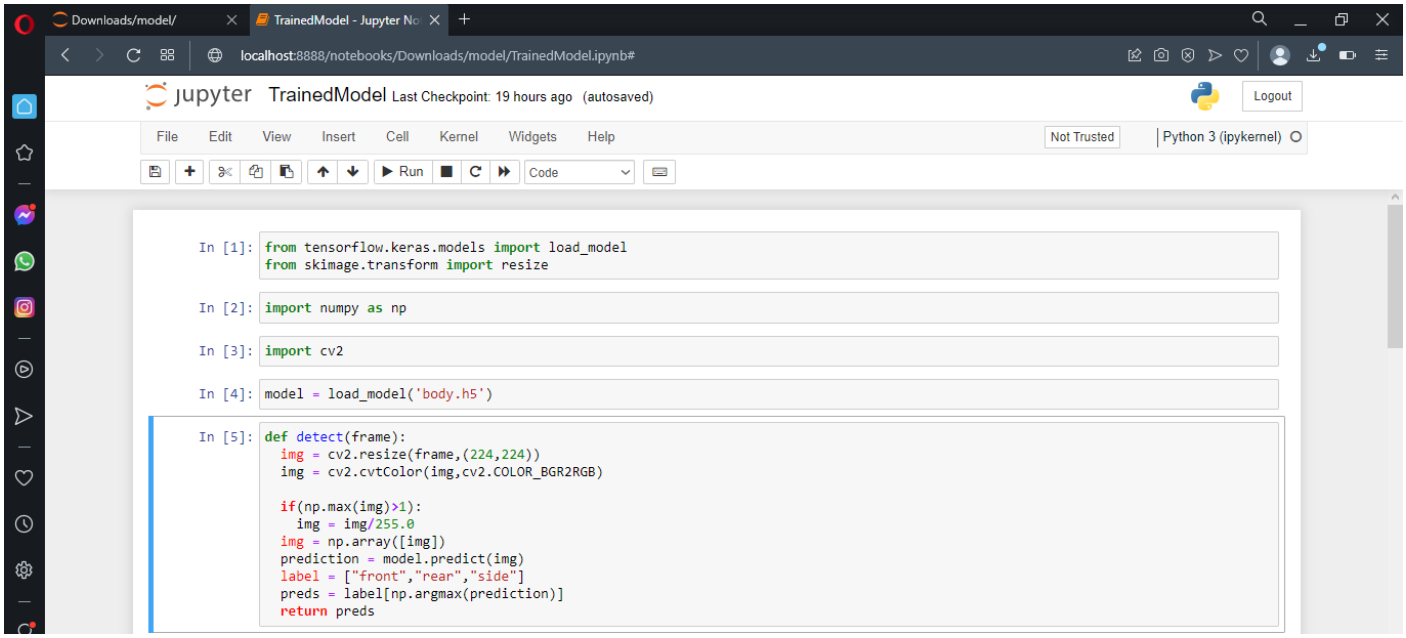
In [ ]:
```
```

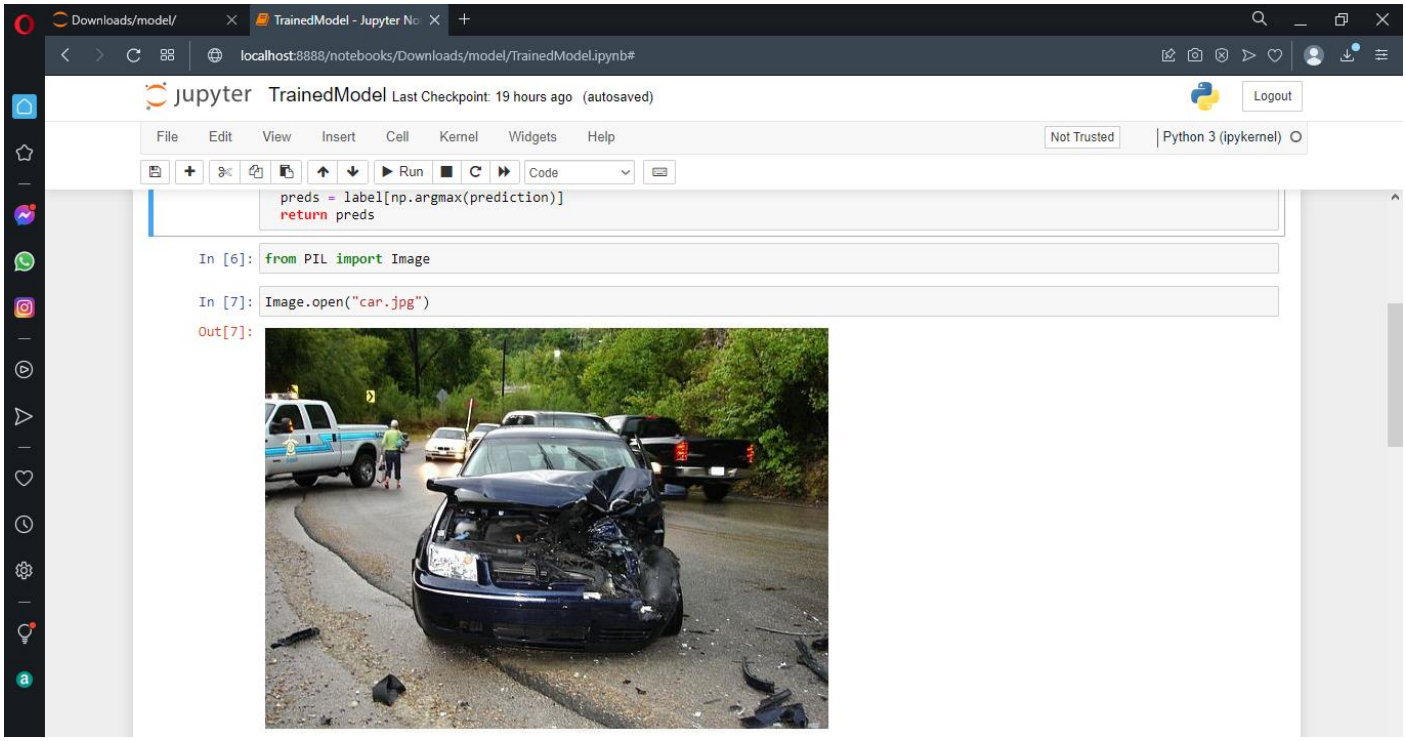
IBM Watson:

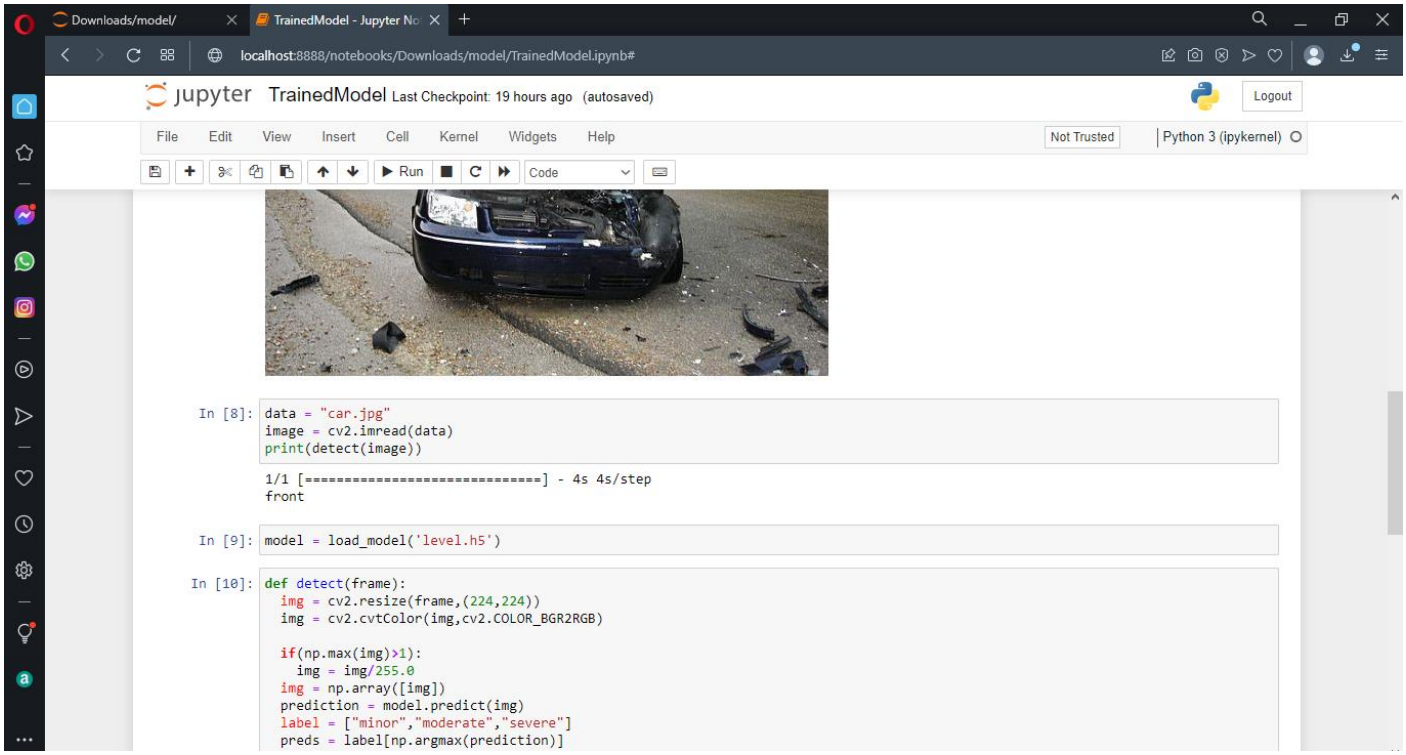

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Watson.ipynb
jupyter IBM Watson Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [1]: !pip install watson-machine-learning-client
Collecting watson-machine-learning-client
Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
Requirement already satisfied: pandas in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (1.4.2)
Requirement already satisfied: tqdm in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (4.64.0)
Requirement already satisfied: lomond in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: tabulate in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: ibm-cos-sdk in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: boto3 in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (1.21.32)
Requirement already satisfied: requests in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (2.27.1)
Requirement already satisfied: certifi in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (2021.10.8)
Requirement already satisfied: urllib3 in c:\users\dell\anaconda3\lib\site-packages (from watson-machine-learning-client) (1.26.9)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in c:\users\dell\anaconda3\lib\site-packages (from boto3->watson-machine-learning-client) (0.5.0)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in c:\users\dell\anaconda3\lib\site-packages (from boto3->watson-machine-learning-client) (0.10.0)
Requirement already satisfied: botocore<1.25.0,>=1.24.32 in c:\users\dell\anaconda3\lib\site-packages (from boto3->watson-machine-learning-client) (1.24.32)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\dell\anaconda3\lib\site-packages (from botocore<1.25.0,>=1.24.32->boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packages (from python-dateutil<3.0.0,>=2.1->botocor

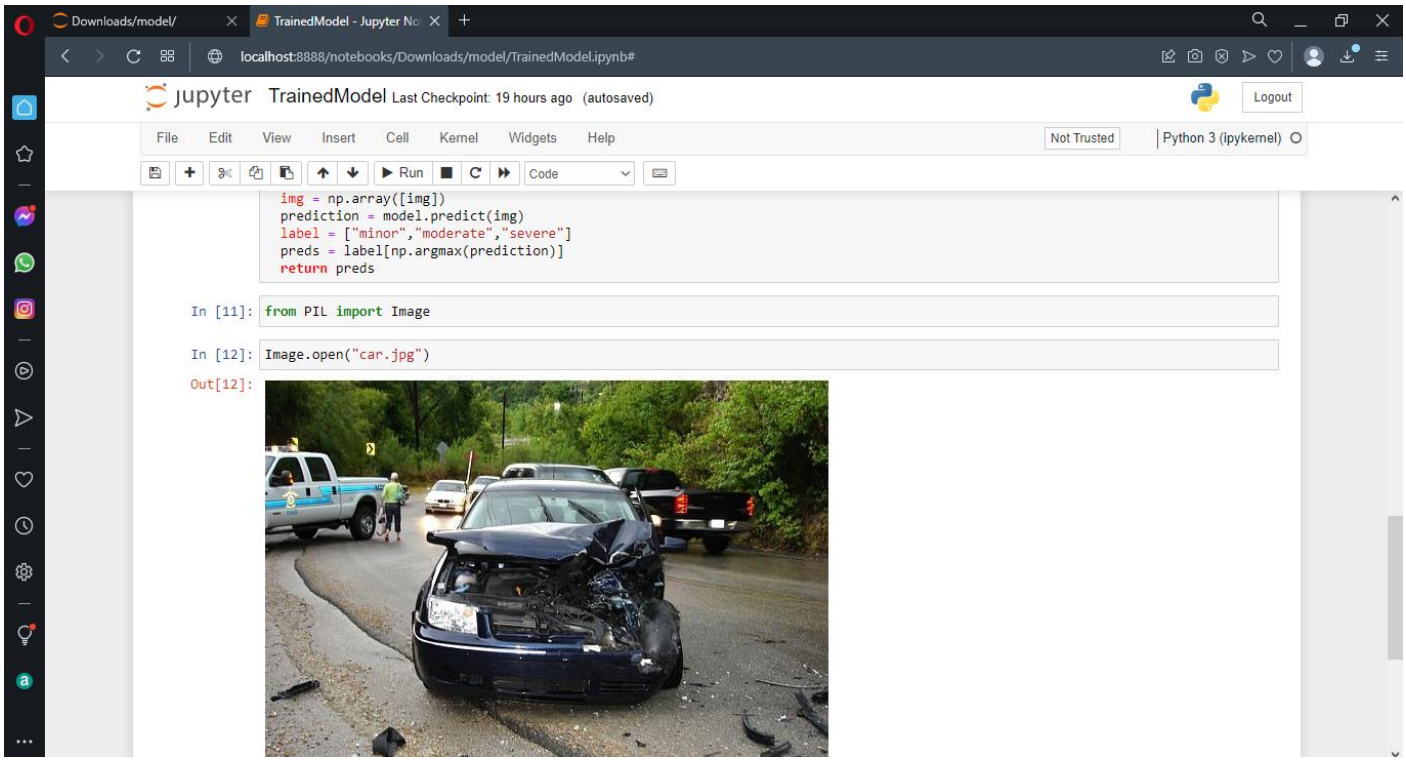

Downloads/
IBM Trained - Jupyter Note
IBM Watson - Jupyter Note
localhost:8888/notebooks/Downloads/IBM%20Watson.ipynb
jupyter IBM Watson Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [2]: from ibm_watson_machine_learning import APIClient
wml_credentials = {
"url": "https://us-south.ml.cloud.ibm.com",
"apikey": "_KrnRAuGk2yTEVQGOZutaRDL-tDvMpwkwtuPWRbEPfnJ"
}
client=APIClient(wml_credentials)
In [3]: client=APIClient(wml_credentials)
In [4]: def guid_from_space_name(client, space_name):
space = client.spaces.get_details()
return(next(item for item in space['resources'] if item['entity']['name'] == space_name)['metadata']['id'])
In [5]: space_uid = guid_from_space_name(client, 'imageclassification')
print("SpaceUID = "+ space_uid)
SpaceUID = a477761e-5761-4737-9e1e-812d7054b2eb
In [6]: client.set.default_space(space_uid)
Out[6]: 'SUCCESS'
In [7]: client.repository.download('361a0dd8-25e7-42c0-a8c7-d15d90530445','model.tar.gz')
Successfully saved model content to file: 'model.tar.gz'
Out[7]: 'C:\\Users\\DELL\\Downloads/model.tar.gz'
In [ ]:
ENG 21:19

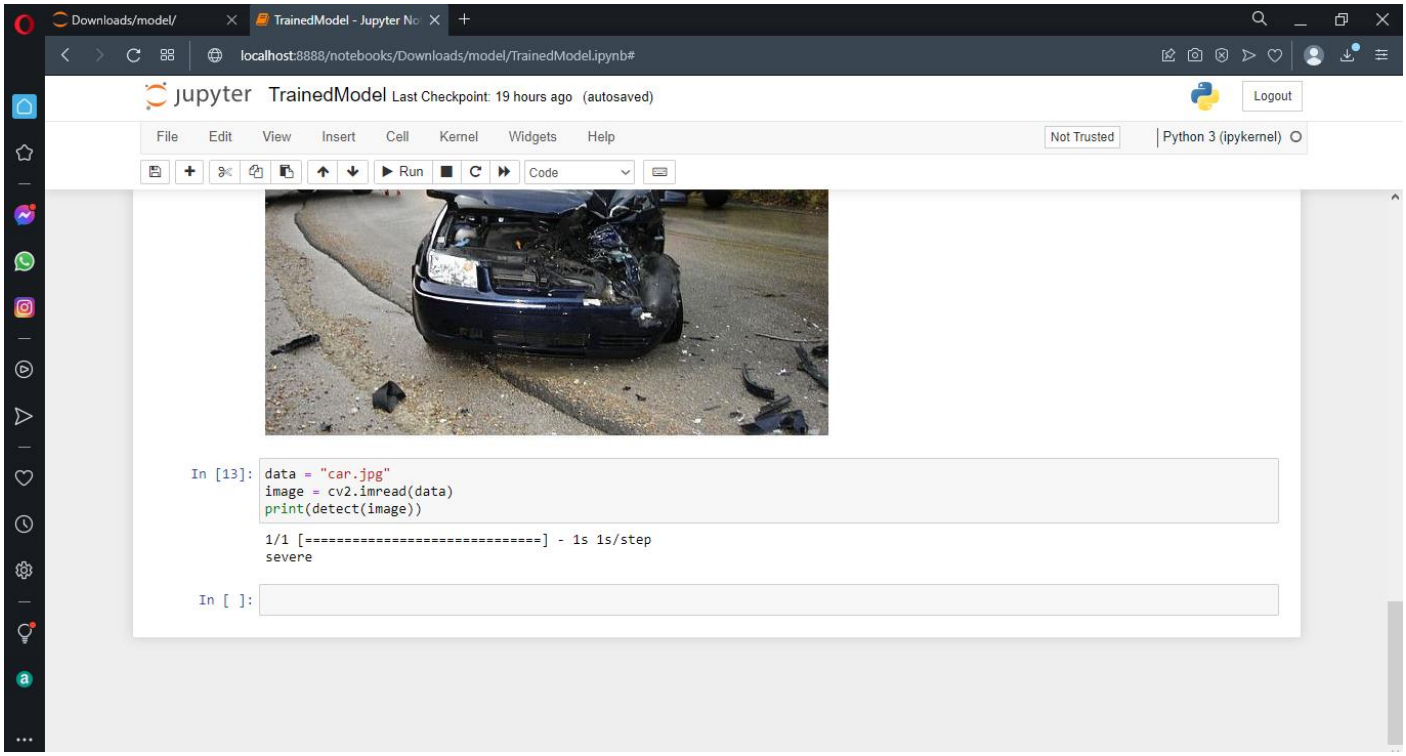
Trained Model:






Downloads/model/
TrainedModel - Jupyter No
localhost:8888/notebooks/Downloads/model/TrainedModel.ipynb#
jupyter TrainedModel Last Checkpoint: 19 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [8]:
data = "car.jpg"
image = cv2.imread(data)
print(detect(image))
1/1 [==============================] - 4s 4s/step
front
In [9]:
model = load_model('level.h5')
In [10]:
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)]


Downloads/model/
TrainedModel - Jupyter No
localhost:8888/notebooks/Downloads/model/TrainedModel.ipynb#
jupyter TrainedModel Last Checkpoint: 19 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
img = np.array([img])
prediction = model.predict(img)
label = ["minor","moderate","severe"]
preds = label[np.argmax(prediction)]
return preds
In [11]:
from PIL import Image
In [12]:
Image.open("car.jpg")
Out[12]:



Frontend:

Index.html

```
<html>
  <head>
    <title>index</title>
    <style type="text/css">

      body{
        background-image: linear-gradient(90deg, #094f79 5%, #00ff84 100%);
      }
      #topmenu {
        width: 100%;
        background-color: 312D2D;
        height: 50px;
      }
      #hedder {
        color: white;
```

```
  padding-top: 13px;
  padding-left: 60px;
}

#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#prediction {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
```

```
  font-size: medium;
}
#about {
  text-align: center;
  padding-top: 10%;
  color: #000000;
  font-size: 20px;
}
#content {
  padding-top: 50px;
  padding-left: 40px;
  padding-right: 40px;
  font-size: large;
}
#footer {
  width: 99%;
  background-color: 312D2D;
  height: 50px;
  position: absolute;
  bottom: 1%;
}
#textcontent {
  color: white;
  font-size: 15px;
  padding-left: 18%;
  padding-top: 1%;
}
#logo {
  margin-top: -1.5%;
  margin-right: 28%;
```

```
        float: right;
      }
    </style>
    <script>
  window.watsonAssistantChatOptions = {
    integrationID: "1d4165da-e3a9-481a-9c00-f93fde429ffe", // The ID of this integration.
    region: "us-south", // The region your integration is hosted in.
    serviceInstanceID: "408f94ed-164a-4206-ab8c-91955c05e343", // The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
 </script>
    </head>
    <body>
    <div id="topmenu">
      <div id="prediction">
        <a href="/prediction" style="color: white;text-decoration: none;">Prediction</a>
      </div>
      <div id="register">
        <a href="/register" style="color: white;text-decoration: none;">Register</a>
      </div>
      <div id="login">
        <a href="/login" style="color: white;text-decoration: none;">Login</a>
      </div>
      <div id="home">
        <a href="#" style="color: white;text-decoration: none;">Home</a>
```

```
    </div>
    <div id="hedder">
      Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance
      Companies
    </div>
  </div>
  <div id="about" >
    PROJECT DETAILS
    <hr style="width: 13%" color="yellow" />
  </div>
  <div id="content">
    <p style="text-align:center">
      Vehicle damage detection is used to reduce claims leakage during
      insurance processing.<br>Visual inception and validation are usually done.
      As it takes a long time, because a person needs to come and inspect the
      damage. <br>Here we are trying to automate the procedure. Using this
      automation, we can avoid time conception for the insurance claim
      problem.
    </p>
  </div>


 </body>
</html>
```

Login.html

```
<html xmlns="http://www.w3.org/1999/html">
 <head>
   <title>Login</title>
```

```
<style type="text/css">
 body{
  background-image: linear-gradient(90deg, #33ccff 5%, #ff99cc 100%);
 }
 #topmenu {
  width: 100%;
  background-color: #1e1e1e;
  height: 50px;
 }
  .picture{
 position:relative;
 left:170px;

 width: 120px;

 }
 #hedder {
  color: white;
  font-size: large;
  padding-top: 13px;
  padding-left: 40px;
 }
 #home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
 }
 #login {
```

```
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#box {
  height: 500px;
  width: 500px;
  background-color: antiquewhite;
  margin: 10px;
  border-color: black;
  border-width: 25px;
}
div.background {
  border: 2px solid gray;
  height: 300px;
  width: 500px;
  margin: auto;
  margin-top: 7%;
}
#loginlogo {
  text-align: center;
```

```
    margin-top: 20px;
   }
   #textcontent {
   margin:10px;
   }
   div.choice {
    border: 2px solid gray;
    height: 35px;
    width: 500px;
    background-color: rgb(230, 227, 227);
    margin: auto;
    margin-top: 0%;
   }

   #question {
    margin-top: 7px;
   }
   #choice-login {
    color: rgb(67, 64, 247);
    text-decoration: underline;
    margin-left: 150px;
    margin-top: -25px;
  }
   .emails{
    top:10px;
   text-align:center;
   }
  </style>
  <script>
window.watsonAssistantChatOptions = {
```

```
      integrationID: "1d4165da-e3a9-481a-9c00-f93fde429ffe", // The ID of this integration.
      region: "us-south", // The region your integration is hosted in.
      serviceInstanceID: "408f94ed-164a-4206-ab8c-91955c05e343", // The ID of your service instance.
      onLoad: function(instance) { instance.render(); }
    };
    setTimeout(function(){
      const t=document.createElement('script');
      t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
      document.head.appendChild(t);
    });
  </script>
  </head>
  <body onload="flashMessage()">
    <div id="topmenu">

      <div id="register">
        <a href="/register" style="color: white;text-decoration: none;">Register</a>
      </div>
      <div id="login">
        <a href="#" style="color: white;text-decoration: none;">Logout</a>
      </div>
      <div id="home">
        <a href="/" style="color: white;text-decoration: none;">Home</a>
      </div>
      <div id="hedder">Login Page</div>

    <!--</div>
    {% with messages = get_flashed_messages() %}
  {% if messages %}
    <ul class=flashes>
```

```
    {% for message in messages %}
      <p><strong>Error:</strong> {{ message }}
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}-->

    <div class="background">
      <div id="loginlogo">
        <lord-icon
        src="https://cdn.lordicon.com/imamsnbq.json"
        trigger="hover"
        style="width:100px;height:100px">
        </lord-icon>
        <!-- <img
          src= "/static/images/login icon.png"
          alt="login logo"
          style="width: 100px; height: 100px; border-radius: 30%"
        /> -->
      </div>
      <div id="textcontent">

        <form action="/afterlogin" method="POST" class="backcolor">
          <script>
            function flashMessage(){
              if("{{flash_message}}" == "True"){
                alert("invalid credentials")
              }
            }
          </script>
```

```
<img src="static/locksmith.png" class=picture align=center >
<input class="emails"
 type="text"
 name="_id"
 id="email"
 placeholder="Enter registered email ID"
 style="width: 440px; height: 35px;  position: relative; left:20px; margin-bottom: 15px; border-radius:20px"
/>

<input class="emails"
 type="password"
 name="psw"
 id="password"
 placeholder="Enter Password"
 style="width: 440px; height: 35px;  position: relative; left:20px; margin-bottom: 25px; border-radius:20px"
/>

<input
 type="submit"
 name="submit"
 value="Login"
 style="
  width: 109px;
  height: 30px;
  text-align: center;
  background-color: black;
  color: white;
 position: relative;
 left:181px;
```

```
                bottom:3px;
            />

    </div>


        </form>

       </div>
     <div class="choice">
       <div id="choice-login">
    <a href="/forgotpassword" style="color: #111111; float:right; padding-right:10px;">Forget password?</a>

       </div>
      </div>
      </div>
      </div>
    </div>
     </body>
    </html>
```

Register.html

```
<html>
  <head>
    <title>Register</title>
    <style type="text/css">
    body{
       background-image: linear-gradient(90deg, #33ccff 5%, #ff99cc 100%);
      }
```

```
#topmenu {
  width: 100%;
  background-color: 312D2D;
  height: 50px;
}
#hedder {
  color: white;
  font-size: large;
  padding-top: 13px;
  padding-left: 40px;
}
#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
```

```
  font-size: medium;
}
#box {
  height: 300px;
  width: 500px;
  background-color: antiquewhite;
  margin: 10px;
  border-color: black;
  border-width: 25px;
}
div.background {
  border: 2px solid gray;
  height: 350px;
  width: 500px;
  margin: auto;
  margin-top: 7%;
}
#registerlogo {
  text-align: center;
  margin-top: 20px;
}
#textcontent {
  margin-top: 28px;
  margin-left: 25px;
}
div.choice {
  border: 2px solid gray;
  height: 35px;
  width: 500px;
  background-color: #000000;
```

```
        margin: auto;
        margin-top: 0%;
       }

       #question {
        margin-top: 7px;
        margin-left: 25;
        color: #ffffff;
       }
       #choice-login {
        color: rgb(67, 64, 247);
        text-decoration: underline;
        margin-left: 225px;
        margin-top: -20px;
       }
      </style>
      <script>
     window.watsonAssistantChatOptions = {
      integrationID: "1d4165da-e3a9-481a-9c00-f93fde429ffe", // The ID of this integration.
      region: "us-south", // The region your integration is hosted in.
      serviceInstanceID: "408f94ed-164a-4206-ab8c-91955c05e343", // The ID of your service instance.
      onLoad: function(instance) { instance.render(); }
     };
     setTimeout(function(){
      const t=document.createElement('script');
      t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
      document.head.appendChild(t);
     });
    </script>
      </head>
```

```
<body onload="flashMessage()">
  <div id="topmenu">
    <div id="login">
      <a href="/login" style="color: white;text-decoration: none;">Login</a>
    </div>
    <div id="home">
      <a href="/" style="color: white;text-decoration: none;">Home</a>
    </div>
    <div id="hedder">Vehicle Damage Detection</div>
  </div>
  <div class="background">
    <div id="registerlogo">
      <img src="static/login.png"
        alt="login logo"
        style="width: 110px; height: 110px; border-radius: 50%"
      />
    </div>
    <div id="textcontent">
      <form action="/afterreg" method="POST">
        <script>
          function flashMessage(){
            if("{{flash_message}}" == "True"){
              alert("account with this email id already exist")
            }

          }
        </script>
        <input
          type="text"
          name="name"
```

```
  id="name"
  placeholder="Enter Name"
  style="border-radius:15px; width: 440px; height: 35px; margin-bottom: 15px"
/>
<input
  type="text"
  name="email"
  id="email"
  placeholder="Enter Email ID"
  style="border-radius:15px; width: 440px; height: 35px; margin-bottom: 15px"
/>
<input
  type="password"
  name="password"
  id="password"
  placeholder="Enter Password"
  style=" border-radius:15px; width: 440px; height: 35px; margin-bottom: 15px"
/>
<input
  type="submit"
  value="Register"
  name="submit"
  style=" border-radius:15px;
    width: 440px;
    height: 35px;
    text-align: center;
    background-color: black;
    color: white;
  "
/>
```

```
      </form>
    </div>
  </div>
  <div class="choice">
    <div id="question">Already Registered?</div>
    <div id="choice-login">
      <a href="/login" style="color: #7ed8ff;">Login</a>
    </div>
  </div>
 </body>
</html>
```

Prediction.html

```
<html>
 <head>
  <title>index</title>
  <style type="text/css">
  body{
     background-image: linear-gradient(90deg, #094f79 5%, #00ff84 100%);
    }
    #topmenu {
     width: 100%;
     background-color: 312D2D;
     height: 50px;
    }
    #hedder {
     color: white;
     padding-top: 13px;
     padding-left: 60px;
```

```
}

#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#prediction {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
```

```
#about {
  text-align: center;
  padding-top: 10%;
  color: gray;
  font-size: 20px;
}
#content {
  padding-top: 50px;
  padding-left: 40px;
  padding-right: 40px;
  font-size: large;
}
#footer {
  width: 99%;
  background-color: 312D2D;
  height: 50px;
  position: absolute;
  bottom: 1%;
}
#textcontent {
  color: white;
  font-size: 15px;
  padding-left: 18%;
  padding-top: 1%;
}
#logo {
  margin-top: -1.5%;
  margin-right: 28%;
  float: right;
}
```

```
    </style>
    <script>
  window.watsonAssistantChatOptions = {
    integrationID: "1d4165da-e3a9-481a-9c00-f93fde429ffe", // The ID of this integration.
    region: "us-south", // The region your integration is hosted in.
    serviceInstanceID: "408f94ed-164a-4206-ab8c-91955c05e343", // The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
 </script>
  </head>
  <body onload="flashMessage()">
    <div id="topmenu">
      <div id="login">
        <a href="/login" style="color: white;text-decoration: none;">Logout</a>
      </div>
      <div id="home">
        <a href="/" style="color: white;text-decoration: none;">Home</a>
      </div>
      <div id="hedder">
        Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance
        Companies
      </div>
    </div>
    <form action="/result" method="POST"  enctype="multipart/form-data">
      <input type="file" id="myFile" name="image">
```

```
  <input type="submit">
  <script>
    function flashMessage(){
      if("{{flash_message}}" == "True"){
      //   alert("invalid credentials")
      //   const im = document.createElement('img');
      // im.src = "{{url_for('static', filename='imagedata/save.png')}}";
      // im.height = "200px";
      // im.width = '200px';
      // im.alt = 'hello world'
      // document.getElementById('about').appendChild(im);
        document.getElementById('image').src = 'static/imagedata/save.png';
        const e = document.getElementById("qwerty");
        const para = document.createElement("p");
        const node = document.createTextNode("The estimated cost for the damage is : | {{value}} |");
        para.appendChild(node);
        e.appendChild(para);
      }
    }
  </script>
</form>
<!-- <script>
  function flashMessage(){
    if("{{ flash_message }}"=='True'){
      const im = document.createElement('img');
      im.src = "{{url_for('static', filename='imagedata/save.png')}}";
      im.height = "200px";
      im.width = '200px';
      im.alt = 'hello world'
    }
```

```
    }
  </script> -->
 <!-- <img src="{{url_for('static', filename='imagedata/save.png')}}" alt=""
  height="200px"
  width="200px"
  /> -->
  <div id="about">
    <div id="qwerty">
      <p></p>
    </div>
    <hr style="width: 30%" color="yellow" />
    {% if prediction %}
       <h3>{{prediction}}</h3>
       <img src={{path}} height="250px" width="400px" alt="" id="image">
    {% endif %}
  </div>



    </body>
</html>
```

Logout.html

```
<html>
 <head>
  <title>Logout</title>
  <style type="text/css">
    #topmenu {
      width: 100%;
      background-color: 312D2D;
```

```
  height: 50px;
}
#hedder {
  color: white;
  font-size: large;
  padding-top: 13px;
  padding-left: 40px;
}
#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#loggedout {
```

```
  color: black;
  font-size: large;
  text-align: center;
  justify-content: center;
  position: absolute;
  top: 50%;
  left: 40%;
  transform: translateY(-500%);
}
#info {
  color: green;
  font-size: small;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  position: absolute;
  top: 50%;
  left: 40%;
  transform: translateY(-500%);
}
#login-button {
  margin: 0%;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  position: absolute;
  top: 50%;
  left: 40%;
```

```
    transform: translateY(-500%0);
   }
  </style>
</head>
<body>
  <div id="topmenu">
  <div id="register">
   <a href="/register" style="color: white;text-decoration: none;">Register</a>
  </div>
  <div id="login">
   <a href="/login" style="color: white;text-decoration: none;">Login</a>
  </div>
  <div id="home">
   <a href="/" style="color: white;text-decoration: none;">Home</a>
  </div>

   <div id="hedder">Vehicle Damage Detection</div>
  </div>
  <div id="loggedout" style="vertical-align: middle">
   Successfully Logged Out!
  </div>
  <div id="info">Login for more information</div>
  <div id="login-button">
   <form action="login">
    <input
     type="submit"
     value="Login"
     style="
      background-color: black;
      color: white;
```

```
            width: 200px;
            height: 35px;
          "
        />
      </form>
    </div>
  </body>
</html>
```

Resetpassword.html

```
<html>
  <head>
    <title>Login</title>
    <script src="https://cdn.lordicon.com/qjzruarw.js"></script>
    <style type="text/css">
      #topmenu {
        width: 100%;
        background-color: 312D2D;
        height: 50px;
      }
      #hedder {
        color: white;
        font-size: large;
        padding-top: 13px;
        padding-left: 40px;
      }
      #home {
        float: right;
        padding-top: 13px;
        padding-right: 50px;
```

```
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#box {
  height: 300px;
  width: 500px;
  background-color: antiquewhite;
  margin: 10px;
  border-color: black;
  border-width: 25px;
}
div.background {
  border: 2px solid gray;
  height: 300px;
  width: 500px;
  margin: auto;
```

```
    margin-top: 7%;
   }
   #loginlogo {
    text-align: center;
    margin-top: 20px;
   }
   #textcontent {
    margin-top: 10px;
    margin-left: 25px;
    margin-top: 20px;
   }
  </style>
</head>
<body onload="flashMessage()">
  <div id="topmenu">
   <div id="register">
    <a href="/register" style="color: white;text-decoration: none;">Register</a>
   </div>
   <div id="login">
    <a href="/login" style="color: white;text-decoration: none;">Logout</a>
   </div>
   <div id="home">
    <a href="/" style="color: white;text-decoration: none;">Home</a>
   </div>
   <div id="hedder">Login Page</div>
  <!--</div>
  {% with messages = get_flashed_messages() %}
{% if messages %}
  <ul class=flashes>
  {% for message in messages %}
```

```
      <p><strong>Error:</strong> {{ message }}
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}-->

    <div class="background">
      <div id="loginlogo">
        <lord-icon
        src="https://cdn.lordicon.com/imamsnbq.json"
        trigger="hover"
        style="width:100px;height:100px">
        </lord-icon>
        <!-- <img
          src= "/static/images/login icon.png"
          alt="login logo"
          style="width: 100px; height: 100px; border-radius: 50%"
        /> -->
      </div>
      <div id="textcontent">

        <form action="resetpassword" method="POST">
          <script>
            function flashMessage(){
              if("{{flash_message}}" == "True"){
                alert("invalid credentials")
              }
            }
          </script>
          <input
```

```
          type="password"
          name="password"
          id="password"
          placeholder="Enter new password"
          style="width: 440px; height: 35px; margin-bottom: 15px"
        />
        <input
          type="submit"
          name="submit"
          value="submit"
          style="
            width: 440px;
            height: 35px;
            text-align: center;
            background-color: black;
            color: white;
          "
        />
      </form>

    </div>
  </div>
 </body>
</html>
```

Backend:

Main.py

```
import os.path
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.models import load_model
import numpy as np
from example import call
from flask import Flask, request, render_template, redirect, url_for
app = Flask(__name__)

my_database,model1, model2 = call()

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

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



@app.route('/afterlogin', methods=['POST'])
def afterlogin():
    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('login.html', pred="The user name is not found")
    else:
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')



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

@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
        '_id': x[1],
        'name': x[0],
        'psw': x[2]
    }
    print(data)
    query = {'_id': {'$eq': 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('register.html', pred="Registration Successful, please login using your details")
    else:
        return render_template('register.html', pred="You are already a member, please login using registered details")


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


@app.route('/forgotpassword')
def forgotpassword():
    return render_template("forgotpassword.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__)
        filepath = os.path.join(basepath,'static',f.filename)
        f.save(filepath)
        img = image.load_img(filepath,target_size=(224,224,1))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis=0)
        img_data = preprocess_input(x)
```

```
model1 = load_model('Model/level.h5')
model2 = load_model('Model/body.h5')
prediction1 = np.argmax(model1.predict(img_data))
prediction2 = np.argmax(model2.predict(img_data))
index1 = ['front', 'rear', 'side']
index2 = ['minor', 'moderate', 'severe']
result1 = index1[prediction1]
result2 = index2[prediction2]

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"
print(value)
```

```
        path='static/'+str(f.filename)
        return render_template('prediction.html', prediction=value,path=path)


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

example.py

```
from tensorflow import keras
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 keras.applications.inception_v3 import preprocess_input
import requests
from flask import Flask, request, render_template, redirect, url_for
from cloudant.client import Cloudant
def call():
    client = Cloudant.iam('b0981d12-395c-4575-8324-ee850683cbde-bluemix','pI9HWcRnB-QJTEhofMuJplmsmaL1QaLbeVAdw1KLge7o',connect=True)
    my_database = client.create_database('my_data')
    model1 = load_model('Model/level.h5')
    model2 = load_model('Model/body.h5')
    return my_database,model1, model2
```

Github Link:

https://github.com/IBM-EPBL/IBM-Project-5272-1658754405

Project Demo Link:

https://github.com/IBM-EPBL/IBM-Project-5272-1658754405/blob/main/Final%20Deliverables/Demo%20Video.mp4

https://ibm-pnt2022tmid01378.s3.jp-tok.cloud-object-storage.appdomain.cloud/Demo%20Video.mp4

https://drive.google.com/file/d/1hIekpBAdsUy8gEYWqscupowxyclN-6Fy/view?usp=share_link