Pallet Analysis

# Technical Design Document

Version 1.0

RESTRICTED DISTRIBUTION

The information is standard Company Confidential, but due to its sensitivity, it has restricted distribution and viewing within iNeuron.

Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | REVIEWER |
| 25th Sep 2020 | 1.0 | Initial Draft | Suresh M |

Contributors

The content of this document has been authored with the combined input of the following group of key individuals.

|  |  |
| --- | --- |
| Name | Section Worked Upon |
| Shilpa Prusty | High level Design , Workflow ,Data Transformers |
| Shubham Kanwal | Data ,model and training configuration, URLs |
| Abhishek Routray | Dockerization |
| Vinod | Monitoring, Logging, Corner cases |
| Vijay Nale |  |
|  |  |
|  |  |

Document Classification

|  |  |
| --- | --- |
| Classification | Company Confidential |
| Definition | Information is Group confidential and needs to be protected |
| Context | Where the loss of information confidentiality would result in significant harm to the interests of the organisation, financial loss, embarrassment or loss of information |

Contents

1[. Introduction 5](#_Toc52215233)

1[.1 High level objectives 5](#_Toc52215234)

2 [Workflow Overall 6](#_Toc52215235)

[Application Flow 6](#_Toc52215236)

[Exception Scenarios Overall 7](#_Toc52215237)

3 [Workflow Data Ingestion and File Conversion 8](#_Toc52215238)

3[.1 Technical solution design 8](#_Toc52215239)

4[. Data Transformers( Pre-processing steps) 9](#_Toc52215240)

4[.1 Technical Design 9](#_Toc52215241)

4[.2 METHOD DEFINITIONS 9](#_Toc52215242)

4[.3 Exceptions Scenarios 11](#_Toc52215243)

5 [Configurations 12](#_Toc52215244)

5[.1 Training Pipeline 12](#_Toc52215245)

5[.2 URLs 12](#_Toc52215246)

5.3 Model Selection Strategy.....................................................................................................12

[6 Testing Modules 19](#_Toc52215251)

[6.1 Technical solution design 19](#_Toc52215252)

[6.2 Exceptions Scenarios Module Wise 19](#_Toc52215253)

[7 Prediction Pipeline 21](#_Toc52215254)

[7.1 Technical solution design 22](#_Toc52215255)

[7.2 Exceptions Scenarios Module Wise 22](#_Toc52215256)

[8 Deployment Strategy 24](#_Toc52215257)

[8.1 Technical solution design 25](#_Toc52215258)

[8.2 Exceptions Scenarios Module Wise 25](#_Toc52215259)

[9 Monitoring 2](#_Toc52215260)9

[9.1 Technical solution design 2](#_Toc52215261)9

[10 Logging](#_Toc52215263) 30

[10.1 Technical solution design](#_Toc52215264) 30

11 Corner cases……………………………………………………………………………………………………………………………31

[from datetime import datetime class App\_Logger: def \_\_init\_\_(self): pass def log(self, file\_object, log\_message):“””This method will be used for logging all the information to the file.””” self.now = datetime.now() self.date = self.now.date() self.current\_time = self.now.strftime("%H:%M:%S") file\_object.write( str(self.date) + "/" + str(self.current\_time) + "\t\t" + log\_message +"\n") 31](#_Toc52215266)

[10.3 Exceptions Scenarios Module Wise 31](#_Toc52215267)

[11 Hardware Requirements 32](#_Toc52215268)

[11.1 Requirements for model training 32](#_Toc52215269)

[11.2 Requirements for model testing 32](#_Toc52215270)

[12 Sample code and standard to be followed: 33](#_Toc52215271)

# Introduction

The current process in place in a warehouse in retil domain is not very effective in detecting missing products as it requires manual check of scanning the box labels with RFID with high error rate . We are building an automated solution of detecting missing boxes in a warehouse by leveraging Deep Learning . We aim to build a pallet and box detection technique using object detection and tracking that can be deployed and served for continuos prediction in real time.

This project shall be delivered in two phases:

Phase 1: All the functionalities with PyPi packages.

Phase2: Integration of UI to all the functionalities along with deployment in cloud/docker container.

The technical design document gives a design blueprint of the Pallet detection project. This document communicates the technical details of the solution proposed.

In addition, this document also captures the different workflows involved to build the solution, exceptions in the workflows and any assumptions that have been considered.

Once agreed as the basis for the building of the project, the flowchart and assumptions will be used as a platform from which the solution will be designed.

Changes to this business process may constitute a request for change and will be subject to the agreed agility program change procedures.

**Note: All the code will be written in python version 3.7**

## 1.1 High level objectives

The high-level objectives are:

* Images collection from various sites by use of web scraping or manual collection of images .
* Perform data preparation by image standardisation [size : 420 \* 420] size each and data augmentation for increasing the training size and model generalization.
* Image storage in MongoDB and count of boxes.
* Choose the appropriate object detection model for training.
* Monitoring training metrics using tensorboard and feeding the logs to logger tool.
* Perform model tuning .
* Making predictions on test images and videos and checking performance.
* Create a list of top 3 models and show multiple metrics for them.
* Give option for prediction and count of each classes detected in the image/video.
* Give options for docker container creation.
* Give option for automatic cloud deployment.

**Phase 1:** Create Pypi packages

**Phase 2:** Integration with Data pipeline for automatic prediction and deployment in container.

# Workflow Overall

## Application Flow

Start

Collect images from various sites

Data augmentation

Image Preprocessing(resize)

Image Annotation (xml format)

Generate tf records

Model training (Object Detection)

Success?

yes

Model Tuning

Validation and Prediction

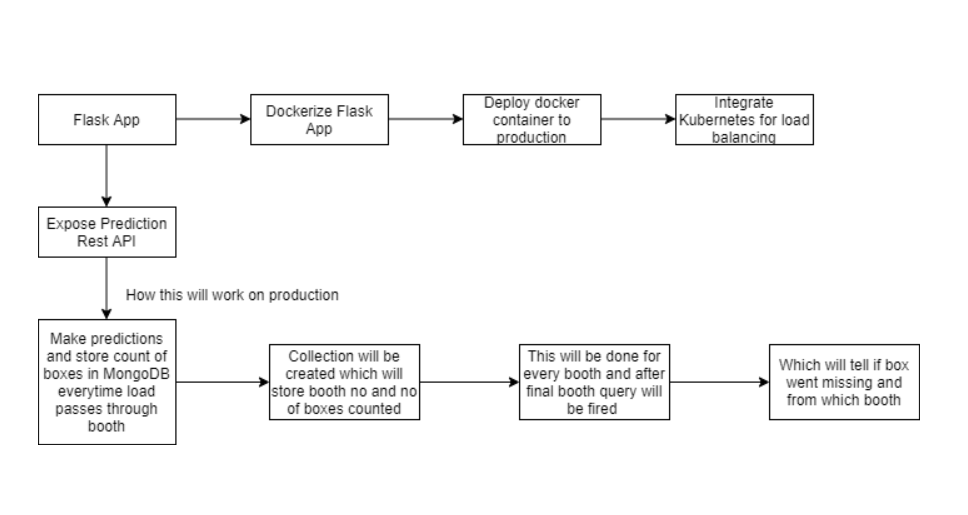
Deployment readiness

No

End

Capture error Message and tuning parametes--Logging

DEPLOYMENT PIPELINE:



## Exception Scenarios Overall

|  |  |  |
| --- | --- | --- |
| **Step** | **Exception** | **Mitigation** |
| User gives Wrong Data Source | Give proper error message | Ask the user to re-enter the details |
| User gives corrupted data | Give proper error message |  |
| Deployment credentials are wrong | Give proper error message | Ask for the details to be entered again |

# Workflow Data Ingestion and File Conversion

**Data Sources:**

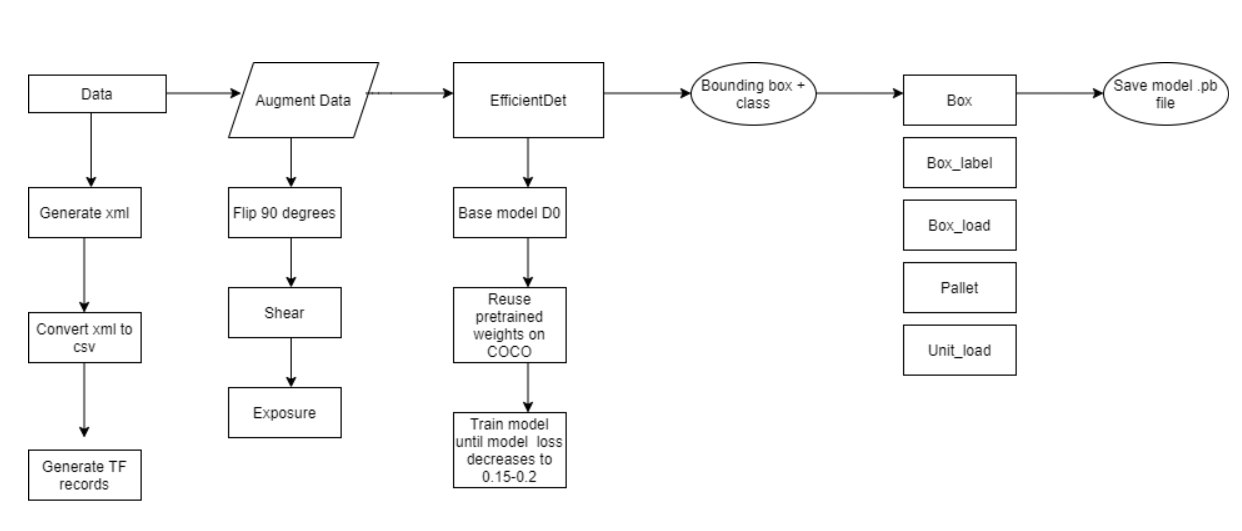
**Phase 1:**

|  |  |
| --- | --- |
| Data Connector Utils | File Conversion Utils |
| Microsoft Office | CSV & text files |
| Labelimg | JSON , XML |
| [Statistical File](https://help.tableau.com/current/pro/desktop/en-us/examples_statfile.htm) | HTML |
| Google Colab | Performance considerations |
| Tensorflow JS |  |
| Roboflow | CSV , jpg |
| MongoDB |  |

**Phase 2:**

|  |  |
| --- | --- |
| Data Connector Utils | File Conversion Utils |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Technical solution design



# 4.Data Transformers( Pre-processing steps)

## 4.1 Technical Design

Collected Data stored in images folder

Augment Data

Image annotation

Image

normalisation

80% 20%

Generate tfrecord

Xml to csv conversion

Train folder

Test folder

Labelmap.pbtxt file creation

unit\_load

boxl

pallet

box

box\_label

## 4.2 METHOD DEFINITIONS

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Data Preprocessor** |  |
| Method Name | rescale\_images |  |
|  | Method Description | Transform image resolution to a common format |
|  | Input Parameter | Directory , size |
|  | Output | Resized images |
|  | On Exception | Show appropriate message if directory cannot be found. |
|  | xml\_to\_csv |  |
|  | Method Description | Convert the xml format consisting of bounding box coordinates to csv format |
|  | Input Parameter | Path to input directory of images |
|  | Output | Returns csv file one for test images and other for train images |
|  | On exception | Log the appropriate error if cannot convert to csv |
|  | Generate\_tf\_record |  |
|  | Method description | Generate tfrecords for train and test images for model training from the csv files. |
|  | Input Parameter | Train or test dir images dir, path to csv containing bounding box coordinates |
|  | Output | Train.tfrecord ;test.tfrecord |
|  | On exception | Log the appropriate error if tfrecords cant be generated |
|  |  |  |
|  | Label\_map\_util |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 4.2Exceptions Scenarios

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# 5. Configuration

5.1. Training Pipeline**-**

**-** Download the model

- This should download a tar.gz file. Once it has downloaded, extracts the contents of the file to the pre-trained-models directory. The structure of that directory should now look something like this

training\_demo/

├─ ...

├─ pre-trained-models/

│ └─ efficientdet\_d0\_coco17\_tpu-32tpu-8/

│ ├─ checkpoint/

│ ├─ saved\_model/

│ └─ pipeline.config

└─ ...

Then open up models\efficientdet\_d0\_coco17\_tpu-32tpu-8\pipeline.config in a text editor because we need to make some changes -

* Line 3. Change num\_classes to 5.
* Line 135. Change batch\_size according to available memory (Higher values require more memory and vice-versa).
* batch\_size: 32
* Line 165. Change fine\_tune\_checkpoint to:
* fine\_tune\_checkpoint: "pre-trained-models/efficientdet\_d0\_coco17\_tpu-32tpu-8/checkpoint/ckpt-0"
* Line 171. Change fine\_tune\_checkpoint\_type to:
* fine\_tune\_checkpoint\_type: "detection"
* Line 175. Change label\_map\_path to:
* label\_map\_path: "annotations/label\_map.pbtxt"
* Line 177. Change input\_path to:
* input\_path: "annotations/train.record"
* Line 185. Change label\_map\_path to:
* label\_map\_path: "annotations/label\_map.pbtxt"
* Line 189. Change input\_path to:
* input\_path: "annotations/test.record"

5.2 URLs

Model url - https://github.com/tensorflow/models

Model weights - 'http://download.tensorflow.org/models/object\_detection/tf2/20200711/

5.3 Model Selection strategy

## Since our use case requires to have a balance between both accuracy and speed.

We filtered out few models.

* SSD Mobilenet
* YOLO v3
* YOLO v4
* EfficientDet-D0

We trained these models using optimized parameters and EfficientDet-D0 gave us the best results on both

images/videos.

# 6 Model Tuning and Optimization

**Note:** The data should have been divided into train and validation set before this.

Methods for hyper tuning all kinds of models.

**Regression:**

Linear Regression

Decision Tree

Random Forest

XG Boost

Support Vector Regressor

KNN Regressor

Model selection criteria:

MSE, RMSE, R squared, adjusted R squared

**Classification:**

Logistic Regression

Decision Tree

Random Forest

XG Boost

Support Vector Classifier

KNN Classifier

Naïve Baye’s

Model selection criteria:

Accuracy, AUC, Precision, Recall, F Beta

**Clustering:**

K-Means

Hierarchial

DBSCAN

**Phase 2:**

GLM

GAM (<https://www.statsmodels.org/stable/regression.html>)

Time Series

Anomaly Detection

Novelty Detection

Optics

Gaussian Mixtures

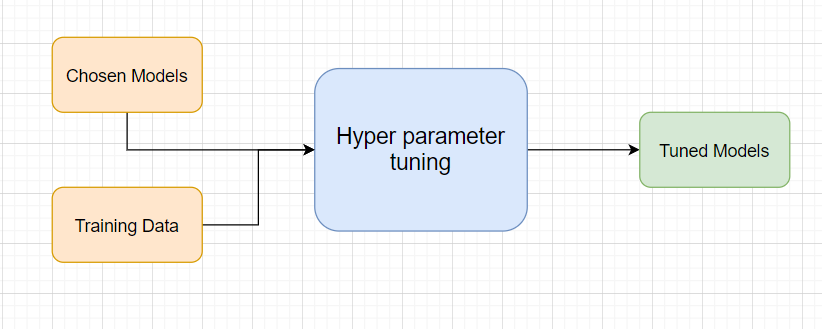
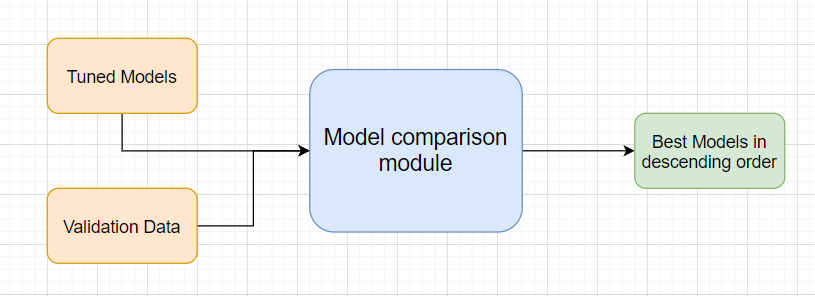
BIRCH

NLP

Deep Learning

Regularization modules if necessary

## Technical solution design

1. 
2. 

## Method Definitions

|  |  |  |
| --- | --- | --- |
| **Class Name** | **ModelTuner** |  |
| Method Name | get\_tuned\_knn\_model |  |
|  | Method Description | This method will be used to get the hypertuned KNN Model |
|  | Input parameter names | self,data |
|  | Input Parameter Description | Data: the training data |
|  | Hyperparameters to tune | **n\_neighbors:**Number of neighbors to use by default for [kneighbors](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#_blank) queries.  **weights:** weight function used in prediction. Possible values:   * ‘uniform’ : uniform weights. All points in each neighborhood are weighted equally. * ‘distance’ : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.   **algorithm**{‘auto’, ‘ball\_tree’, ‘kd\_tree’, ‘brute’}, default=’auto’  Algorithm used to compute the nearest neighbors:  **leaf\_size:** int, default=30  Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.  **n\_jobs:** int, Keep it as -1 |
|  | ouptput | A hyper parameter tuned model object |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |
| Method Name | get\_tuned\_random\_forest\_model |  |
|  | Method Description | This method will be used to get the hypertuned Random Forest Model |
|  | Input parameter names | self,data |
|  | Input Parameter Description | Data: the training data |
|  | Hyperparameters to tune | **Classifier**  **n\_estimators:** The number of trees in the forest.  **criterion**{“gini”, “entropy”}, default=”gini”  The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.  **max\_depth**: int, default=None  The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.  **min\_samples\_split:** int or float, default=2  The minimum number of samples required to split an internal node:  **n\_jobs=** -1  **Regressor**  **n\_estimators:** The number of trees in the forest.  **criterion**{“mse”, “mae”}, default=”mse”  The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.  **max\_depth**: int, default=None  The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.  **min\_samples\_split:** int or float, default=2  The minimum number of samples required to split an internal node: |
|  | ouptput | A hyper parameter tuned model object |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |
| Method Name | get\_tuned\_xgboost\_model |  |
|  | Method Description | This method will be used to get the hypertuned XGBoost Model |
|  | Input parameter names | self,data |
|  | Input Parameter Description | Data: the training data |
|  | Hyperparameters to tune | eta [default=0.3, alias: learning\_rate]   * Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative. * range: [0,1]   gamma [default=0, alias: min\_split\_loss]   * Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. * range: [0,∞]   max\_depth [default=6]   * Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 is only accepted in lossguided growing policy when tree\_method is set as hist and it indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree.   Objective: The objective function |
|  | ouptput | A hyper parameter tuned model object |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |

## Exceptions Scenarios Module Wise

|  |  |  |
| --- | --- | --- |
| **Step** | **Exception** | **Mitigation** |
|  |  |  |

# 7 Testing Modules

Divide the training data itself into train and test sets

Use test data to have tests run on the three best models

Give the test report

1. R2 Score
2. Adjusted R2 score
3. MSE
4. Accuracy
5. Precision
6. Recall
7. F Beta
8. Cluster Purity
9. Silhouette score

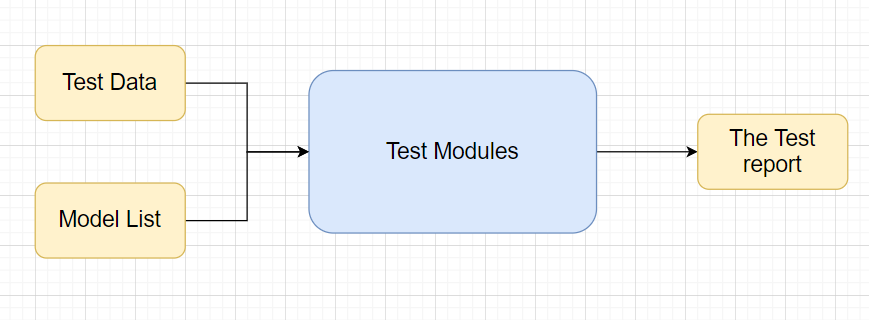
**Phase 2**

AIC

BIC

**Note**: Save the best model after validation is completed.

## Technical solution design



## Exceptions Scenarios Module Wise

|  |  |  |
| --- | --- | --- |
| **Step** | **Exception** | **Mitigation** |
| Number of Parameters do not match | Handle internally | Check the test data creation and verify the columns |
| Only once class present in test data | Handle Internally |  |

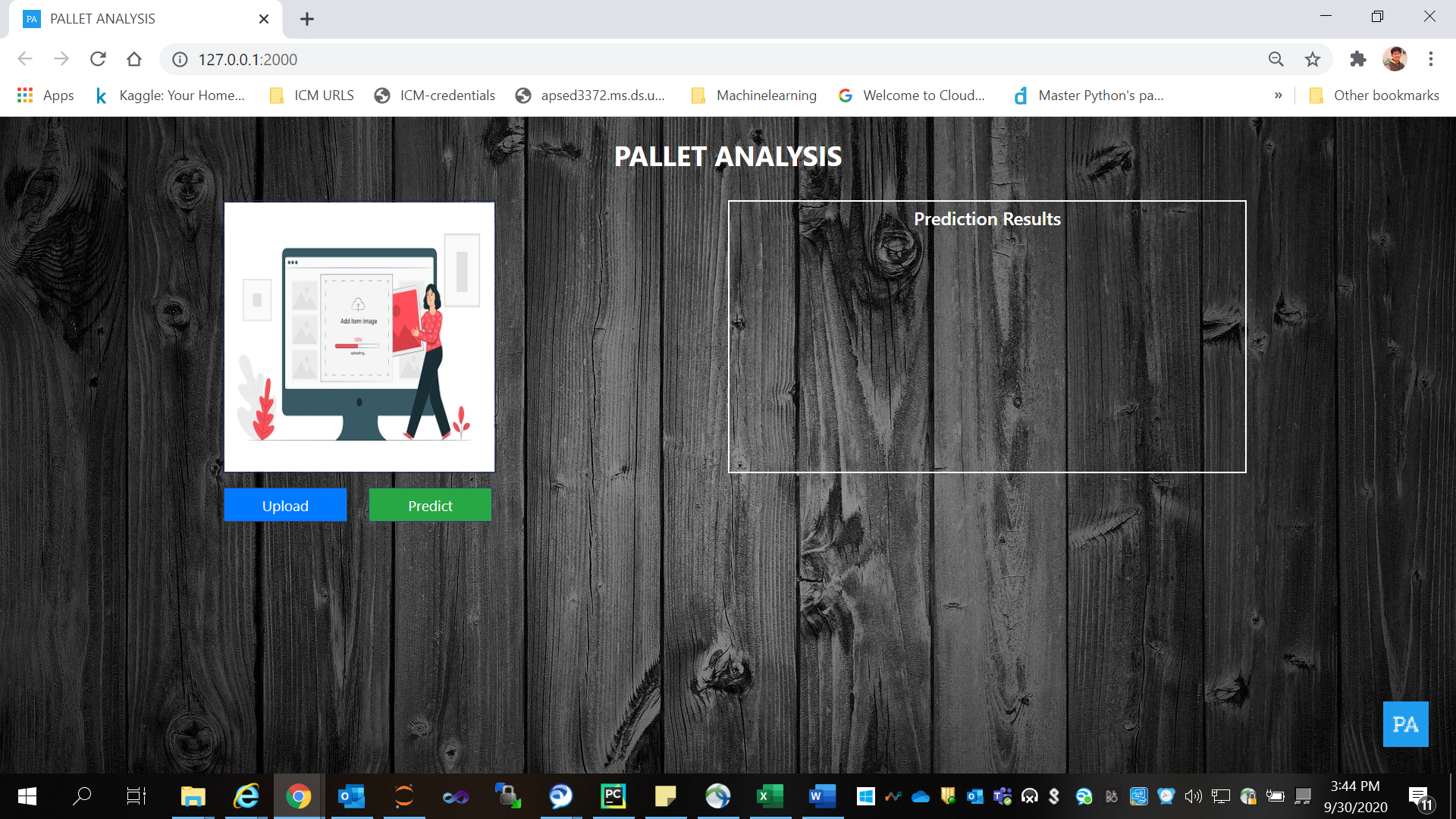
# 8 Prediction Pipeline

**Steps for prediction**

1. **Use the existing data read modules**
2. **Use the existing pre-processing module**
3. **Load the model into memory**
4. **Do predictions**
5. **Store prediction results(show sample predictions)**

**Phase 2:**

**UI for predictions**



**User End Flow:**

**1.Click on upload button on the UI**

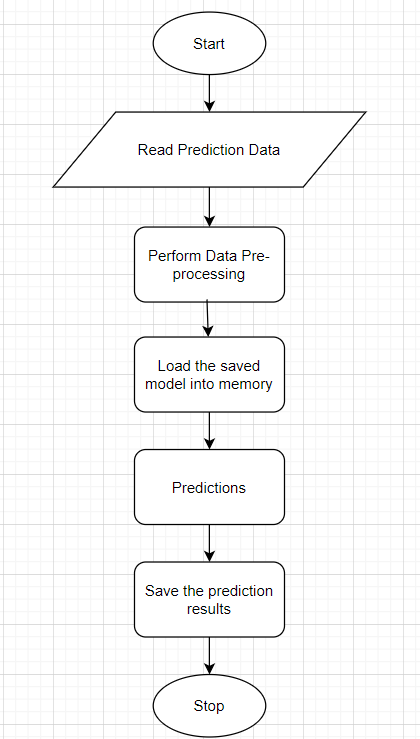
**2.Select image from your local directory**

**3.check image preview on UI**

**4.Click predict**

**5.Check prediction result under prediction results**

## **Design Flow**



## Exceptions Scenarios Module Wise

|  |  |  |
| --- | --- | --- |
| **Step** | **Exception** | **Mitigation** |
| Valid input data format | Show error message as “Upload Valid format data” | The user enters the correct data format |
| If input is invalid | Show error message as “Upload Valid data” | The user enters the correct data |

# 8 . Deployment Strategy

Before Deployment A Dockerfile is created in the project Directory.

10 steps to deploy on kubernetes engine :

## **Step 1 — Create a new project in GCP Console.**

## **Step 2 — Import Project Code**

Click the **Activate Cloud Shell**button at the top of the console window to open the Cloud Shell.

Execute the following code in Cloud Shell to clone the GitHub repository used in this project. Git clone <https://github.com/shubhamkanwal/Pallet_detection.git>

## **Step 3— Set Project ID Environment Variable**

Execute the following code to set the PROJECT\_ID environment variable.

export PROJECT\_ID=pallet-analysis-demo

Pallet-analysis-demo is the name of the project we chose in step 1 above.

## **Step 4— Build the docker image**

docker build -t gcr.io/${PROJECT\_ID}/pallet-app:v1 .

## **Step 5— Upload the container image**

1. Authenticate to [Container Registry](https://cloud.google.com/container-registry) (you need to run this only once):

gcloud auth configure-docker

1. Execute the following code to upload the docker image to Google Container Registry:

docker push gcr.io/${PROJECT\_ID}/pallet-app:v1

## **Step 6— Create Cluster**

Now that the container is uploaded, you need a cluster to run the container. A cluster consists of a pool of Compute Engine VM instances, running Kubernetes.

1. Set your project ID and Compute Engine zone options for the gcloud tool:

gcloud config set project $PROJECT\_ID

gcloud config set compute/zone **us-central1**

2. Create a cluster by executing the following code:

gcloud container clusters create pallet**-cluster** --num-nodes=2

## **Step 7— Deploy Application**

To deploy and manage applications on a GKE cluster, you must communicate with the Kubernetes cluster management system. Execute the following command to deploy the application:

kubectl create deployment pallet-app --image=gcr.io/${PROJECT\_ID}/pallet-app:v1

## **Step 8— Expose your application to the internet**

By default, the containers you run on GKE are not accessible from the internet because they do not have external IP addresses. Execute the following code to expose the application to the internet:

kubectl expose deployment pallet-app --type=LoadBalancer --port 80 --target-port 8080

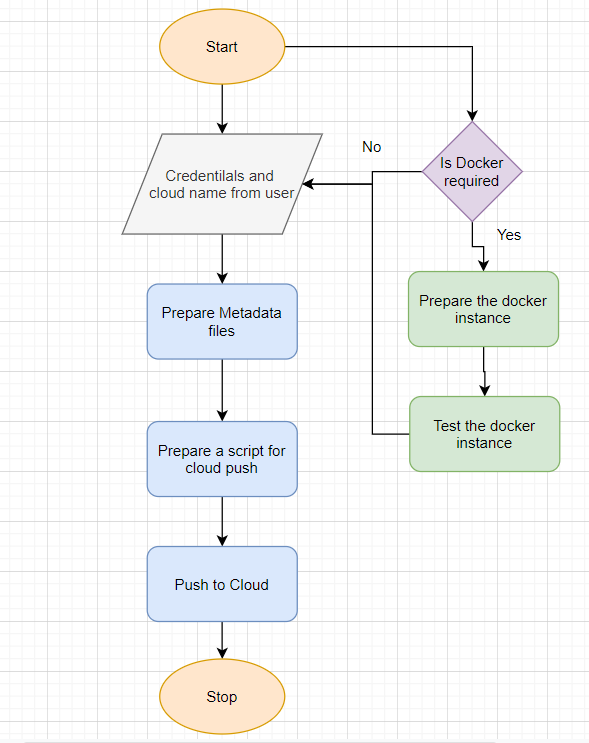
## **Step 9— Check Service**

Execute the following code to get the status of the service. **EXTERNAL-IP** is the web address you can use in browser to view the published app.

kubectl get service

**Step 10—** See the app in action on [http://**EXTERNAL-IP**:8080](http://34.71.77.61:8080/)

## Technical solution design



## Exceptions Scenarios Module Wise

|  |  |  |
| --- | --- | --- |
| **Step** | **Exception** | **Mitigation** |
| Wrong Cloud credentials | Show error message | The user enters the correct data |
| Docker instance not working | Show error message | Fix the error |
| Cloud push failed | Show the error | Make corrections to the metadata  files |
| Cloud app not starting |  | Ask the user for cloud logs for debugging |

# Monitoring

## 9.1 Technical solution design

* **Scalars** - Visualize scalar values, such as classification accuracy.
* **Graph** - Visualize the computational graph of your model, such as the neural network model.
* **Histograms** - A fancier view of the distribution that shows distributions in a 3-dimensional perspective
* **Distributions** - Visualize how data changes over time, such as the weights of a neural network.

# Logging

Separate Folder for logs

Logging of every step

Entry to the methods

Error message Logging

## 10.1 Technical solution design



|  |  |  |
| --- | --- | --- |
| Method Name | logging |  |
|  | Method Description | This method will be used for logging all the information to the file. |
|  | Input parameter names | self, log\_file, log\_dir, log\_level, log\_message. |
|  | Input Parameter Description | log\_file: name of the log file  log\_dir: directory of the log file  log\_level: the type of severity that is written in the log\_file.  log\_message: the message to be logged |
|  | output | A log file with messages |
|  | On Exception | Ideally, the logging should never fail. |

# Corner cases

NEED TO BE ADDED

# Hardware Requirements

## Requirements for model training

The minimum configuration should be:

* 8 GB RAM
* 2 GB of Hard Disk Space
* Intel Core i5 Processor

## Requirements for model testing

The minimum configuration should be:

* 4 GB RAM
* 2 GB of Hard Disk Space
* Intel Core i5 Processor

# Sample code and standard to be followed:

Sample Code:

class Data\_Getter:  
 *"""  
 This class shall be used for obtaining the data from the source for training.  
  
 Written By: iNeuron Intelligence  
 Version: 1.0  
 Revisions: None  
  
 """* def \_\_init\_\_(self, file\_object, logger\_object):  
 self.training\_file='Training\_FileFromDB/InputFile.csv'  
 self.file\_object=file\_object  
 self.logger\_object=logger\_object  
  
 def get\_data(self):  
 *"""  
 Method Name: get\_data  
 Description: This method reads the data from source.*

*Input Description:   
 Output: A pandas DataFrame.  
 On Failure: Raise Exception  
  
 Written By: iNeuron Intelligence  
 Version: 1.0  
 Revisions: None  
  
 """* self.logger\_object.log(self.file\_object,'Entered the get\_data method of the Data\_Getter class') # Logging entry to the method  
 try:  
 self.data= pd.read\_csv(self.training\_file) # reading the data file  
 self.logger\_object.log(self.file\_object,'Data Load Successful.Exited the get\_data method of the Data\_Getter class') # Logging exit from the method  
 return self.data # return the read data to the calling method  
 except Exception as e:  
 self.logger\_object.log(self.file\_object,'Exception occured in get\_data method of the Data\_Getter class. Exception message: '+str(e)) # Logging the exception message  
 self.logger\_object.log(self.file\_object,  
 'Data Load Unsuccessful.Exited the get\_data method of the Data\_Getter class') # Logging unsuccessful load of data  
 raise Exception() # raising exception and exiting

Coding Standard:

1. Imports should usually be on separate lines
2. Avoid trailing whitespace anywhere. Because it's usually invisible, it can be confusing.
3. Compound statements (multiple statements on the same line) are generally discouraged
4. Comments should be complete sentences. Always make a priority of keeping the comments up-to-date when the code changes. Ensure that your comments are clear and easily understandable to other speakers of the language you are writing in.
5. Never use the characters 'l' (lowercase letter el), 'O' (uppercase letter oh), or 'I' (uppercase letter eye) as single character variable names.
6. The name of the variables should start with small case capital letters and a multi word variable should be named as: word1\_word2\_word3.
7. The variable name should be appropriate based on the things that they do. DO NOT USE NAMES LIKE x, k, y etc. Always use a meaningful English word. For example, customer\_name, nearest\_neighbour etc.
8. Method names should start with small case characters. They should start with a verb and make a meaningful sense of what they are supposed to accomplish. For e.g.: load\_data\_from\_sql()
9. Always use self for the first argument to instance methods.
10. Class names should normally use the CapWords convention. Class name should also represent the functionality of the class. For e.g. DataLoader()
11. Modules/Packages/Folders should have short, all-lowercase names. Underscores can be used in the module name if it improves readability. For e.g.: data\_ingestion
12. Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples include MAX\_OVERFLOW and TOTAL.
13. Comparisons to singletons like None should always be done with is or is not, never the equality operators
14. The code should be properly enclosed withing try and exception blocks and the exceptions should be handled with proper error messages.
15. Additionally, for all try/except clauses, limit the try clause to the absolute minimum amount of code necessary. Again, this avoids masking bugs
16. When a resource is local to a particular section of code, use a with statement to ensure it is cleaned up promptly and reliably after use.
17. Be consistent in return statements. Either all return statements in a function should return an expression, or none of them should. If any return statement returns an expression, any return statements where no value is returned should explicitly state this as return None, and an explicit return statement should be present at the end of the function (if reachable)
18. Object type comparisons should always use isinstance() instead of comparing types directly
19. Don't compare boolean values to True or False using ==