Project 1: Automated Passenger Board Kiosk

Introduction

In this document, the solution for Project 1 in the Udacity AI Engineer with Microsoft program is described. The contents of this report try to meet the criteria stated in the course website. All my solutions were submitted to a personal github repository and the steps and reasoning followed can be tracked in the issues section I created to manage and organize the project.

Preparate Project Materials

Flight Manifest Table

Can be found in my github repo for this project: manifest table.

Boarding Passes

I generated 5 boarding passes to analyze per the minimum requirements (to create a custom form model I created an extra 10 boarding passes, which will be shown in following sections). The boarding passes live all in blob storage in Azure, but here, to make the review easier, in Fig. 1 the boarding passes are depicted.

Digital Ids

I used the California driver's license provided in the materials to create five ids, matching the fake persons in the boarding passes, such IDs are shown in fig. 2.

UDACITY AIRLINES

Passenger Name	Carrier	Flight No.	Class	Passenger Name
Norma Vásquez	LATAM	LA-21	Tourist	Norma Vásquez
			From	Concepción
			To	Santiago
From: Concepción	Date	Baggage	Seat	Seat	Date
To: Santiago	12 April 2022	176950	10F	10F	12 April 2022
GATE	Boarding Time			GATE	Boarding Time
A5	8:00 AM				

UDACITY AIRLINES

Passenger Name	Carrier	Flight No.	Class	Passenger Name
Manuel Gaete	SKY	SKY70	Tourist	Manuel Gaete
			From	Concepción
			To	Vallenar
From: Concepción	Date	Baggage	Seat	Seat	Date
To: Vallenar	21 March 2022	112323	20C	20C	21 March 2022
GATE	Boarding Time			GATE	Boarding Time
G20	2:00 PM				

UDACITY AIRLINES

Passenger Name	Carrier	Flight No.	Class	Passenger Name
Javiera Strogonoff	LATAM	LA-23	Tourist	Javiera Strogonoff
			From	Arica
			To	Punta Arenas
From: Arica	Date	Baggage	Seat	Seat	Date
To: Punta Arenas	18 December 2022	B894274	20E	20E	18 December 2022
GATE	Boarding Time			GATE	Boarding Time
B1	5:00 PM				

UDACITY AIRLINES

Passenger Name	Carrier	Flight No.	Class	Passenger Name
Diogo Puelma	LATAM	LA-21	First	Diogo Puelma
			From	Santiago
			To	Concepción
From: Santiago	Date	Baggage	Seat	Seat	Date
To: Concepción	10 April 2022	I1234123	3A	3A	11 August 2021
GATE	Boarding Time			GATE	Boarding Time
A5	11:00 AM				

UDACITY AIRLINES

Passenger Name	Carrier	Flight No.	Class	Passenger Name
Clynton Tomacheski	Jet Smart	Jet3	Tourist	Clynton Tomacheski
			From	Temuco
			To	Puerto Montt
From: Temuco	Date	Baggage	Seat	Seat	Date
To: Puerto Montt	25 February 2022	I86950	35B	35B	25 February 2022
GATE	Boarding Time			GATE	Boarding Time
D7	8:00 AM				

Fig. 1. Boarding passes used for the project

Fig. 2. Fake document IDs used for the project.

Azure Resources and Storage

In fig. 3, azure resources and blob storage is shown. In particular, the manifest was uploaded to the blob storage. The reason for this, is because for deployment purposes, in the case of having a server running on azure, it has to be access to the manifest for checks (and the manifest in reality might change over time, for example on a daily basis).

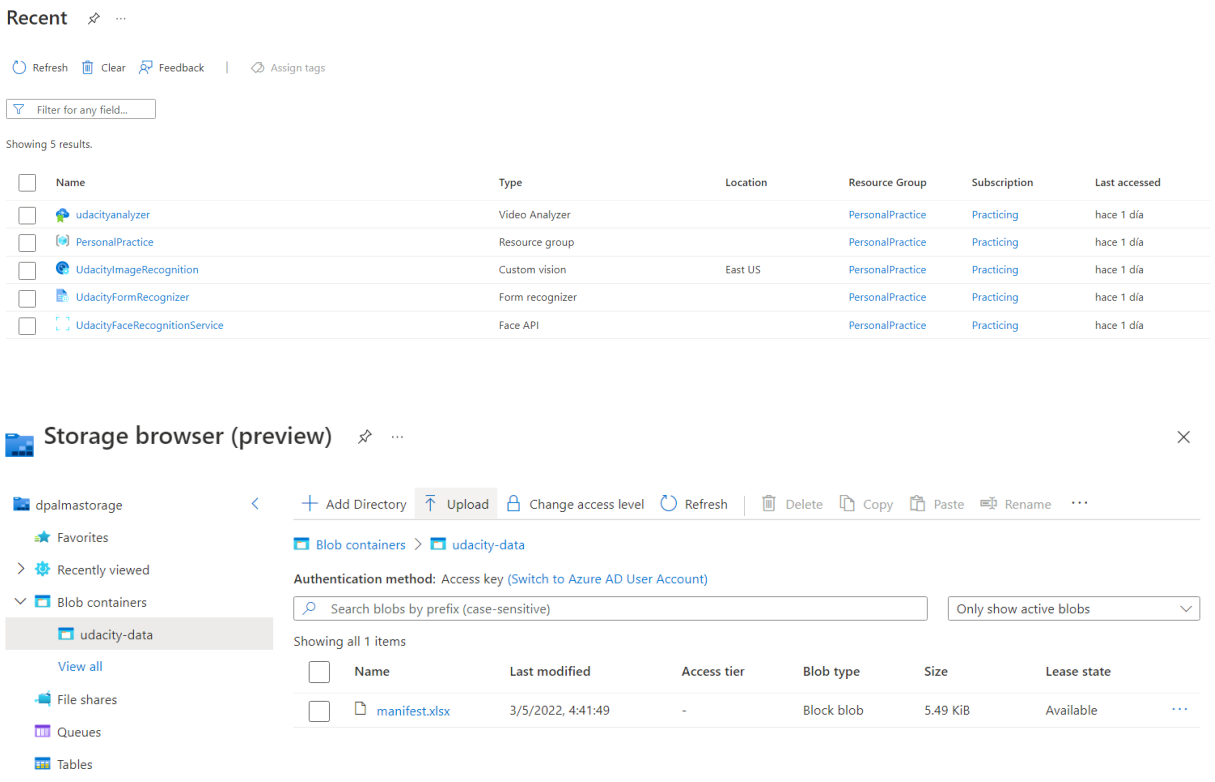

Fig. 3. Azure resources and showing how blob storage was used.

For experiment purposes, I uploaded two videos, one using glasses and one without glasses, which are shown on fig. 4.

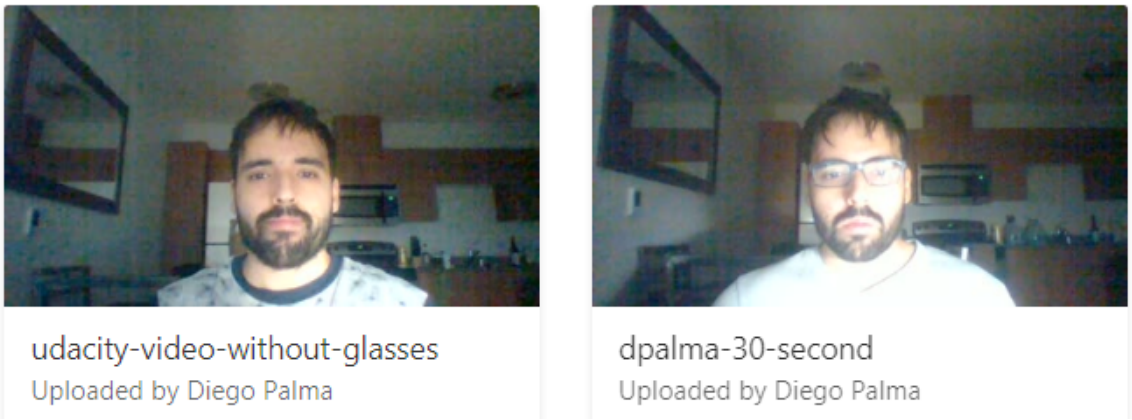

Fig. 4. Videos submitted to VideoIndexer

Problem Definition

In an airport, validating a person's data is a task that currently is done manually. The process usually takes a long time, in some cases causing a bad customer experience. Moreover, since it is a human performed task, it might be prone to errors, and resources could not be used efficiently (e.g. causing long wait times). There are several checks that need to be performed, such as:

- Check the identity of the person who carries the boarding pass
- Check that the boarding pass is legit
- Checking carry on items (this is the longest task)

On the other hand, it could be useful for an airport and an airline to get insights about customer experience. Usually, this data is obtained through surveys, which are also time consuming.

Solution Strategy

For the problems mentioned above, a possible solution to make the processes efficient is by automating repetitive tasks, and this could be achieved by using computer vision approaches. For example, boarding pass data could be automatically extracted from a picture and compared with a customer identification card. Carry on items might be pre-checked before passing through the conveyor belt, analyzing it via cameras and using object detection techniques to detect dangerous items. Finally, for customer experience analytics, customer sentiments could be extracted from their facial expressions.

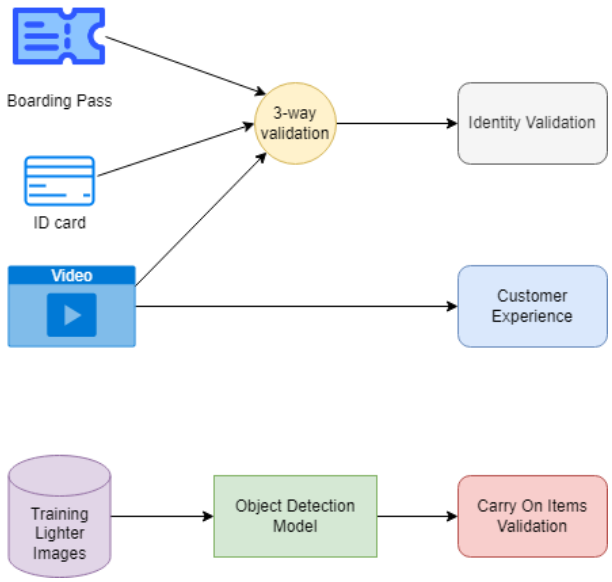

Fig. 5. Flow for automated passenger board kiosk

Data Sources

In Fig. 1, it is shown the flow of the described problem. As for the solution described early, it can be summarized as follows: *Leveraging repetitive tasks by automating them using computer vision technologies:*

- Person Identity validation:
  - Video data
  - Identification Card data (e.g. driver's license)
  - Boarding Pass
- Customer Experience automation
  - Analyzing faces from customers and extracting insights such as emotions and sentiment based
- Dangerous Object Detection on carry on items:
  - Checking a photo of carry on items and detecting dangerous items (e.g. lighter)

Cognitive Services

We have two alternatives for implementing such solutions:

1. Build all the infrastructure by scratch, and developing all the technologies from scratch
2. Use services from a cloud provider, for example Microsoft Azure, which has cognitive services that could be used to implement the described solution.

In table 1, candidate services to be used are shown.

Table 1. Cognitive Services used to implement the solution

Data Processed	Microsoft Cognitive Service
Identification Document Processing	Azure Form Recognizer
Boarding Pass Processing	Azure Custom Form Recognizer
Vídeo Analysis	Microsoft Video Indexer
Face Recognition	Azure Face API service
Object Detection	Azure Custom Vision service

Data Validation

To validate data for a person identity, two validations where done:

- A 3-way validation using boarding pass information, identification document information and manifest table information.
- Compare faces from the identification document with faces detected in the video camera. For this, a confidence threshold might be used, and for the project a threshold of 0.65 or higher was chosen to determine if the person in the video is the person in the id.

To validate data from carry on items, a computer vision detection model was trained using image samples and labeling the dangerous objects. To make sure the model quality was good enough, precision, recall and mean average precision were used as metrics. Then, the model was deployed and could be used for predictions on new images.

Proposed Architecture

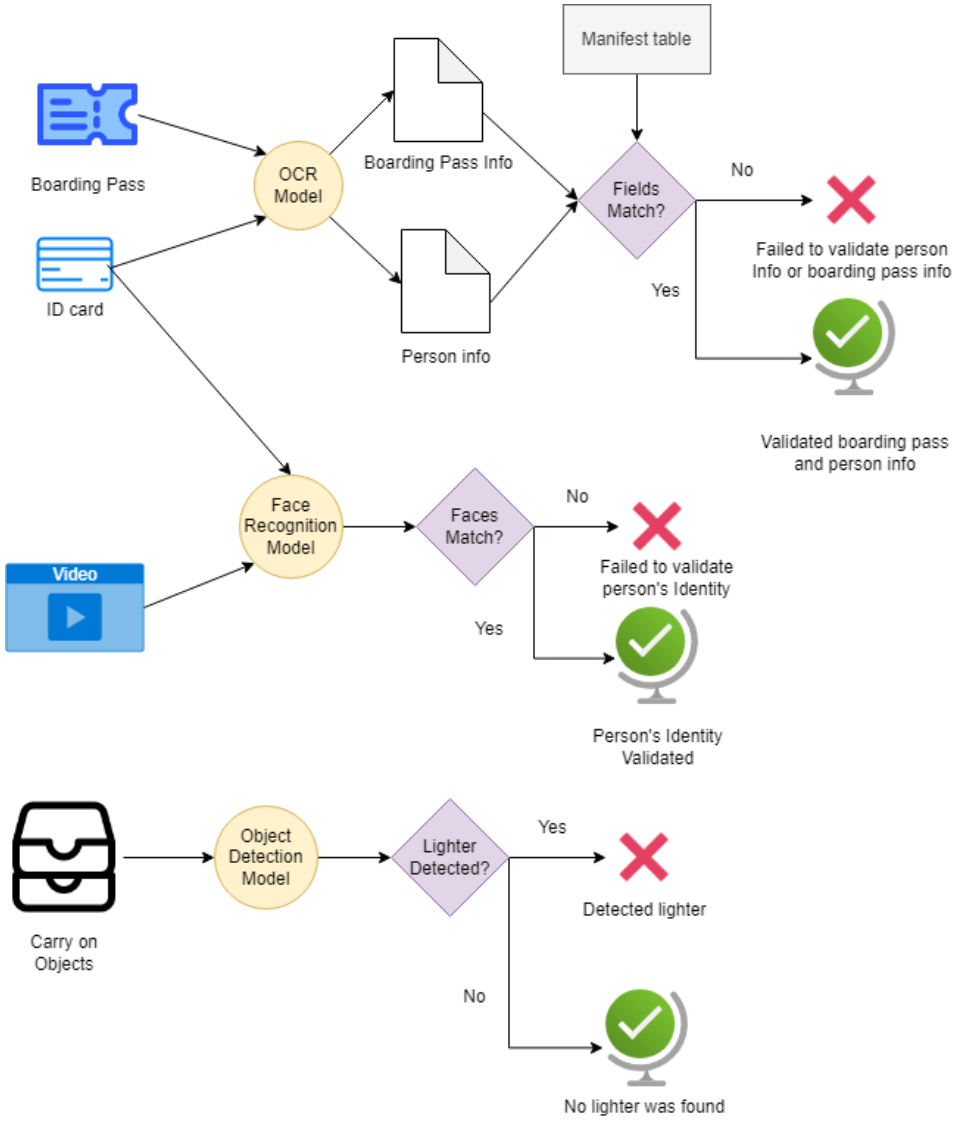

Fig. 6. Flow for automated passenger board kiosk.

In fig. 6 it is depicted the data flow architecture and the components that are needed to build the described solution. On the other hand, in fig. 7, it is shown the diagram containing the Microsoft Services to be used to build such components.

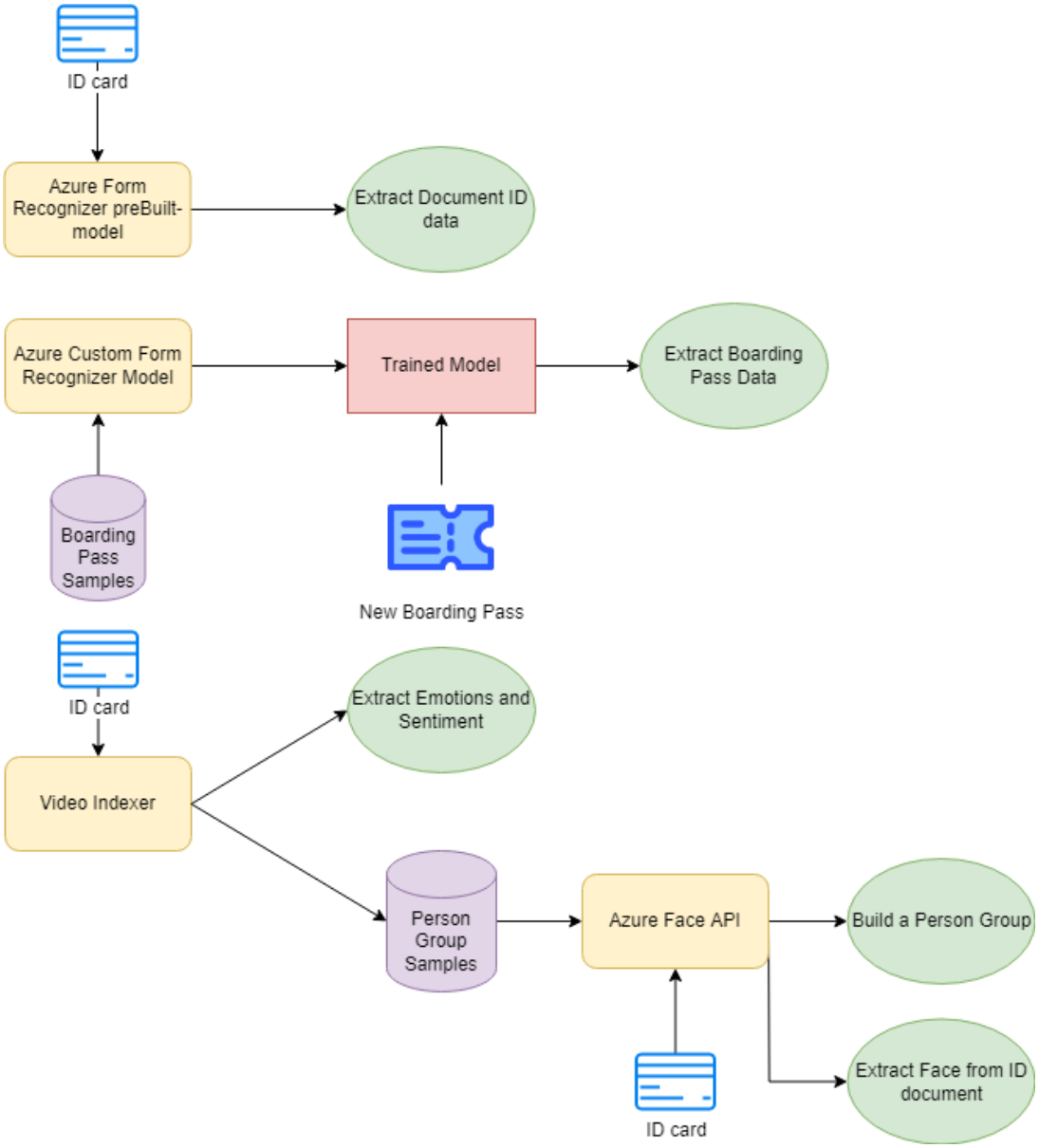

Fig. 7. Building components with cognitive services.

Text Data Extraction

Collect Data from ID images using Form Recognizer's prebuilt ID API

Running the code in snippet 1 on python in debug mode (check config to set log_level) results as the output shown in snippet 2. Note that the function receives as input two Recognizer objects, which were custom objects I implemented to process the data using Form Recognizer REST API.

Snippet 1. Python code to get form data from Form Recognizers

```
def get_form_info(
    id_doc_model: Recognizer,
    boarding_pass_model: Recognizer,
    person_id_doc: str,
    person_boarding_pass: str,
):
    logger.debug(f"Will process {person_id_doc} {person_boarding_pass}")
    resource = f"{BLOB_STORAGE_URL}/{person_id_doc}"
    op_location = id_doc_model.analyze(resource)
    id_info = id_doc_model.result(op_location)
    id_relevant_info = {}
    for field, value in id_info["analyzeResult"]["documents"][0][
        "fields"
    ].items():
        if field in MANIFEST_MAPPINGS:
            id_relevant_info[MANIFEST_MAPPINGS[field]] = (
                value["valueDate"]
                if value["type"] == "date"
                else value["valueString"]
            )
    resource = f"{BLOB_STORAGE_URL}/{person_boarding_pass}"
    op_location = boarding_pass_model.analyze(resource)
    boarding_pass_info = boarding_pass_model.result(op_location)
    boarding_pass_relevant_info = {}
    for field, value in
boarding_pass_info["analyzeResult"]["documents"][0][
        "fields"
    ].items():
        if field in MANIFEST_MAPPINGS:
            field = MANIFEST_MAPPINGS[field]
            boarding_pass_relevant_info[field] = (
                value["valueDate"]
                if value["type"] == "date"
                else process_boarding_value(field, value["valueString"])
            )

    logger.debug(f"Id document info: {id_relevant_info}")
    logger.debug(f"Boarding pass info: {boarding_pass_relevant_info}")
    return id_relevant_info, boarding_pass_relevant_info
```

Snippet 2. Output of running snippet 1 in DEBUG mode.

```
2022-05-07 15:42:02,566 main  DEBUG Manifest data: {'diogo': {'Flight No':
'LA-21', 'Origin': 'Santiago', 'Destination': 'Concepción', 'Date': '10
April 2022', 'Time': '11:00', 'First Name': 'Diogo Andrés', 'Last Name':
'Puelma', 'Sex': 'M', 'SeatNo': '3A', 'DateOfBirth': '17 January 1990',
'DoBValidation': 'FALSE', 'PersonValidation': 'FALSE', 'LuggageValidation':
'FALSE', 'NameValidation': 'FALSE', 'BoardingPassValidation': 'FALSE'},
'manuel': {'Flight No': 'SKY70', 'Origin': 'Concepción', 'Destination':
'Vallenar', 'Date': '21 March 2022', 'Time': '14:00', 'First Name': 'Manuel
Ignacio', 'Last Name': 'Gaete', 'Sex': 'M', 'SeatNo': '20C', 'DateOfBirth':
'3 March 1989', 'DoBValidation': 'FALSE', 'PersonValidation': 'FALSE',
'LuggageValidation': 'FALSE', 'NameValidation': 'FALSE',
'BoardingPassValidation': 'FALSE'}, 'clynton': {'Flight No': 'Jet3',
'Origin': 'Temuco', 'Destination': 'Puerto Montt', 'Date': '25 February
2022', 'Time': '8:00', 'First Name': 'Clynton Augusto', 'Last Name':
'Tomacheski', 'Sex': 'M', 'SeatNo': '35B', 'DateOfBirth': '6 May 1992',
'DoBValidation': 'FALSE', 'PersonValidation': 'FALSE', 'LuggageValidation':
'FALSE', 'NameValidation': 'FALSE', 'BoardingPassValidation': 'FALSE'},
'norma': {'Flight No': 'LA-21', 'Origin': 'Concepción', 'Destination':
'Santiago', 'Date': '12 April 2022', 'Time': '8:00', 'First Name': 'Norma
Alejandra', 'Last Name': 'Vásquez', 'Sex': 'F', 'SeatNo': '10F',
'DateOfBirth': '5 September 1987', 'DoBValidation': 'FALSE',
'PersonValidation': 'FALSE', 'LuggageValidation': 'FALSE',
'NameValidation': 'FALSE', 'BoardingPassValidation': 'FALSE'}, 'javiera':
{'Flight No': 'LA-23', 'Origin': 'Arica', 'Destination': 'Punta Arenas',
'Date': '18 December 2022', 'Time': '17:00', 'First Name': 'Javiera
Alejandra', 'Last Name': 'Strogonoff', 'Sex': 'F', 'SeatNo': '20E',
'DateOfBirth': '15 November 1994', 'DoBValidation': 'FALSE',
'PersonValidation': 'FALSE', 'LuggageValidation': 'FALSE',
'NameValidation': 'FALSE', 'BoardingPassValidation': 'FALSE'}}
2022-05-07 15:42:02,577 main  DEBUG Will process ca-dl-diogo.png
boarding_pass_diogo.pdf
2022-05-07 15:42:10,619 main  DEBUG Id document info: {'DateOfBirth':
'1990-01-17', 'First Name': 'Diogo Andrés', 'Sex': 'M'}
2022-05-07 15:42:10,619 main  DEBUG Boarding pass info: {'Time': '11:00',
'Last Name': 'Puelma', 'Date': '10 April 2022', 'Origin': 'Santiago',
'Destination': 'Concepción', 'First Name': 'Diogo', 'SeatNo': '3A', 'Flight
No': 'LA-21'}
```

Note that the information extracted from the ID photo matches the actual information in the photo, as you might notice by looking at fig. 8.

Fig. 8. Sample ID document from where data was extracted using prebuilt Form Recognizer.

Build custom boarding passes recognizer model and extract data from boarding passes

Ten boarding passes samples were used to train a custom recognizer model. Fig. 1 shows the training step, fig. 2 shows the test phase and fig. 3 shows the training boarding passes in a blob storage bucket.

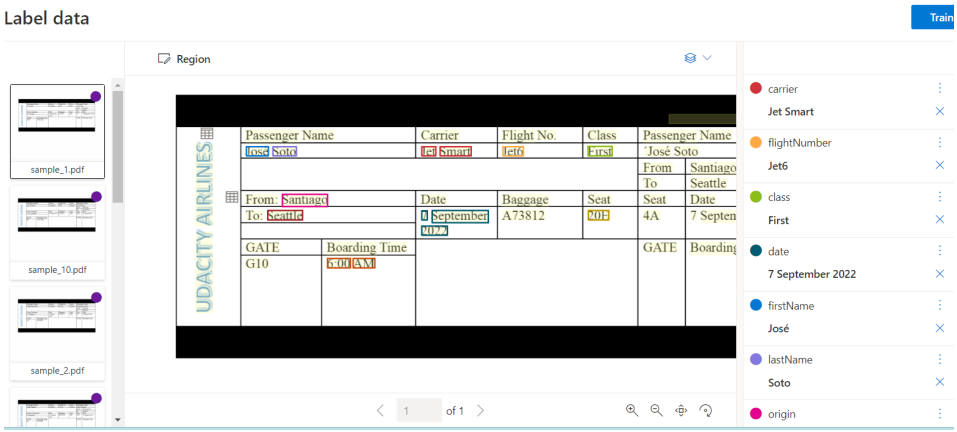

Fig. 7. Building components with cognitive services.

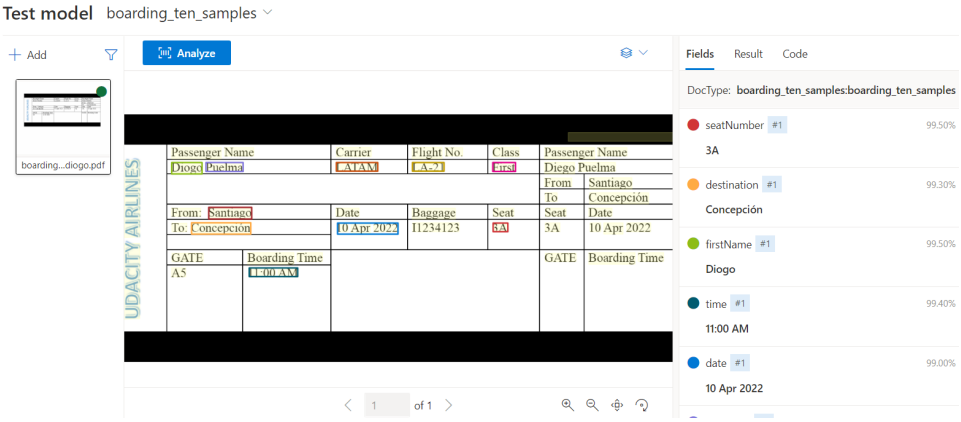

Fig. 8. Checking the same boarding pass as in snippet 2.

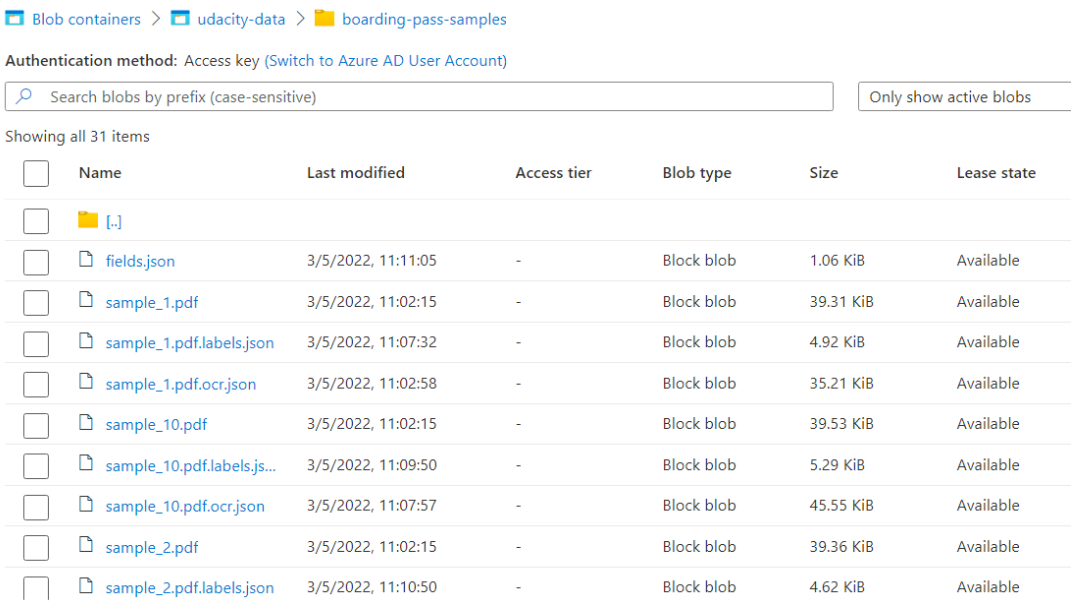

Fig. 9. Training samples and labels in an Azure blob storage.

I used the GUI to train the model, as it was more intuitive and faster to develop. The python code recognizing the form data is in snippet 1 (the same for the prebuilt model as both use the same API).

Face Data Extraction

Snippet 3. Uploading a video to videoindexer.

```
def upload_video(indexer: VideoIndexer) -> str:
    uploaded_video_id = indexer.upload_to_video_indexer(
        input_filename="/mnt/c/Users/dpalma/Desktop/ids/udacity-video.mp4",
        video_name="dpalma-30-second",
        video_language="English",
    )
    return uploaded_video_id
```

Snippet 4. Show image thumbnail.

```
from pathlib import Path
from PIL import Image
import matplotlib.pyplot as plt

ID_PATH = Path("/mnt/c/Users/dpalma/Desktop/ids")
with open(ID_PATH / "human-face1.JPG", "rb") as img_code:
    img_view_ready = Image.open(img_code)
    _, ax = plt.subplots()
    ax.imshow(img_view_ready)

plt.show()
```

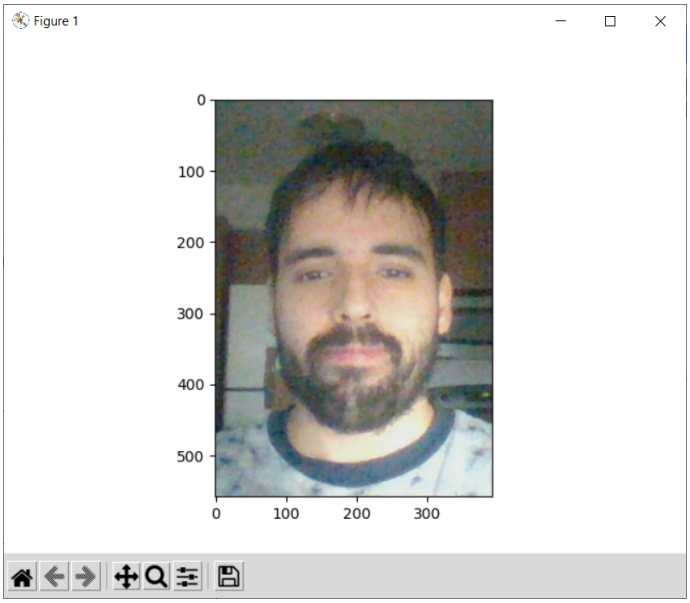

Fig. 9. Output of snippet 4.

Snippet 5. Build the person model from thumbnails.

```
def build_person_group(
    client: FaceClient,
    person_group_id,
    pgp_name,
    path: Path,
    prefix="human-face",
):
    logger.debug("Create and build a person group...")
    # Create empty Person Group. Person Group ID must be lower case,
```

```
    # alphanumeric, and/or with '-', '_'.
    logger.debug(f"Person group ID: {person_group_id}")
    client.person_group.create(
        person_group_id=person_group_id, name=person_group_id
    )

    # Create a person group person.
    human_person = client.person_group_person.create(person_group_id,
pgp_name)
    # Find all jpeg human images in working directory.
    human_face_images = [
        file
        for file in glob.glob(str(path / "*.jpg"))
        if os.path.basename(file).startswith(prefix)
    ]
    # Add images to a Person object
    for image_p in human_face_images:
        with open(image_p, "rb") as w:
            try:
                client.person_group_person.add_face_from_stream(
                    person_group_id, human_person.person_id, w
                )
            except Exception as e:
                if config.is_free_tier:
                    logger.debug(f"{e}, will try to wait a minute")
                    time.sleep(60)
                    client.person_group_person.add_face_from_stream(
                        person_group_id, human_person.person_id, w
                    )
                else:
                    raise

    # Train the person group, after a Person object with many images
    # were added to it.
    client.person_group.train(person_group_id)

    # Wait for training to finish.
    while True:
        training_status = client.person_group.get_training_status(
            person_group_id
        )
        logger.debug("Training status: {}.".format(training_status.status))
```

```
        if training_status.status is TrainingStatusType.succeeded:
            break
        elif training_status.status is TrainingStatusType.failed:
            client.person_group.delete(person_group_id=person_group_id)
            sys.exit("Training the person group has failed.")
        time.sleep(5)
```

Snippet 6. Read the person's face from the ID document.

```
image_path = id_path / person_id_doc
logger.debug("Reading image {image_path}")
with open(image_path, "rb") as img:
    try:
        dl_faces = face_client.face.detect_with_stream(img)
    except Exception as e:
        if config.is_free_tier:
            logger.debug(f"{e} Free tier, will wait a minute")
            time.sleep(60)
            dl_faces = face_client.face.detect_with_stream(img)
        else:
            raise

for face in dl_faces:
    dl_id = face.face_id
```

Custom Object Detection

The following information is shown in the next set of figures:

- Fig. 10, fig. 11 and fig. 12: Uploading and labeling training samples
- Fig. 13: Finishing training the model with the training samples and performance metrics of the model
- Fig. 14: Publishing the model to an endpoint (this could be done using the REST API) or python SDK client. It was faster for me just using the GUI
- Snippet 7: Making predictions with the model using the endpoint
- Fig. 15: Test images and recognized objects in bounding boxes

Regarding figure 15 I added an extra image with no lighter and set the threshold to 0.95, however the model still thinks that there is a lighter in the photo. Even though the model performance metrics look good, we cannot be deceived and we need to check corner cases with new test samples to actually verify the performance on out of sample data. A possible fix to this model would be reducing the overlap threshold for bounding boxes, and relabel the data with complex cases.

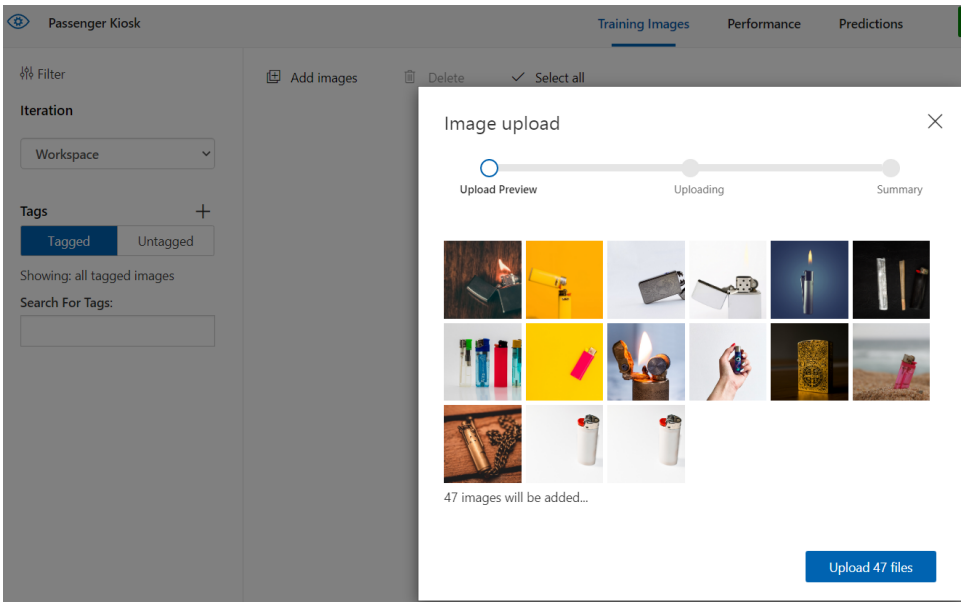

Fig. 10. Uploading training images.

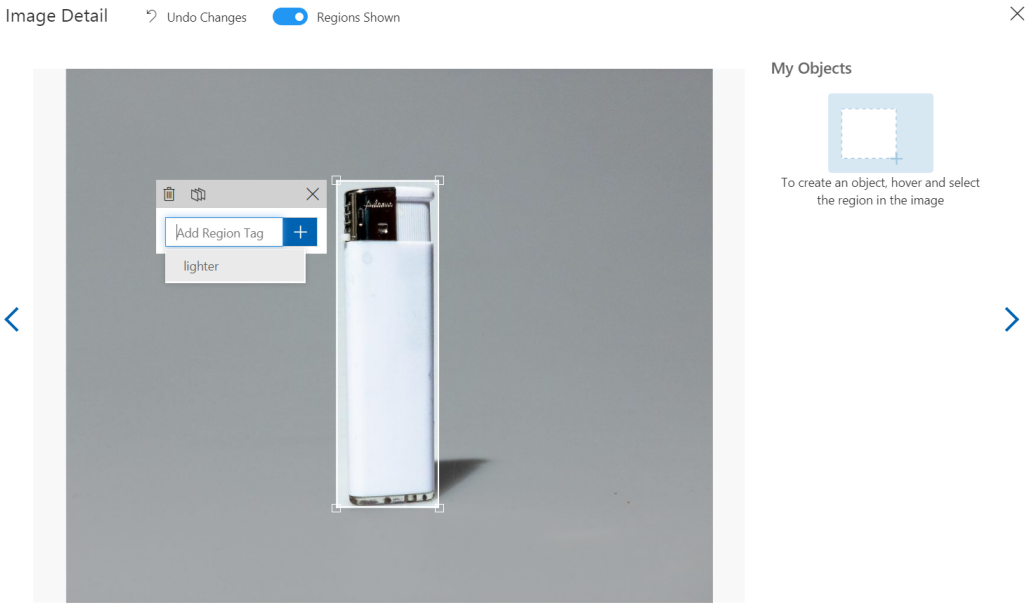

Fig. 11. Creating a label.

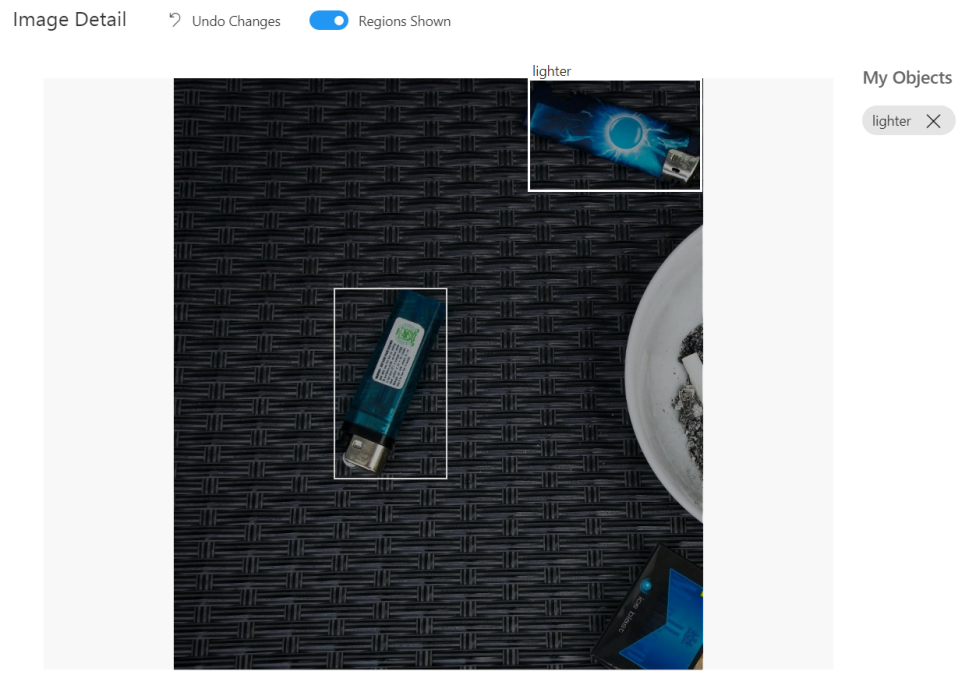

Fig. 12. Labeling training images.

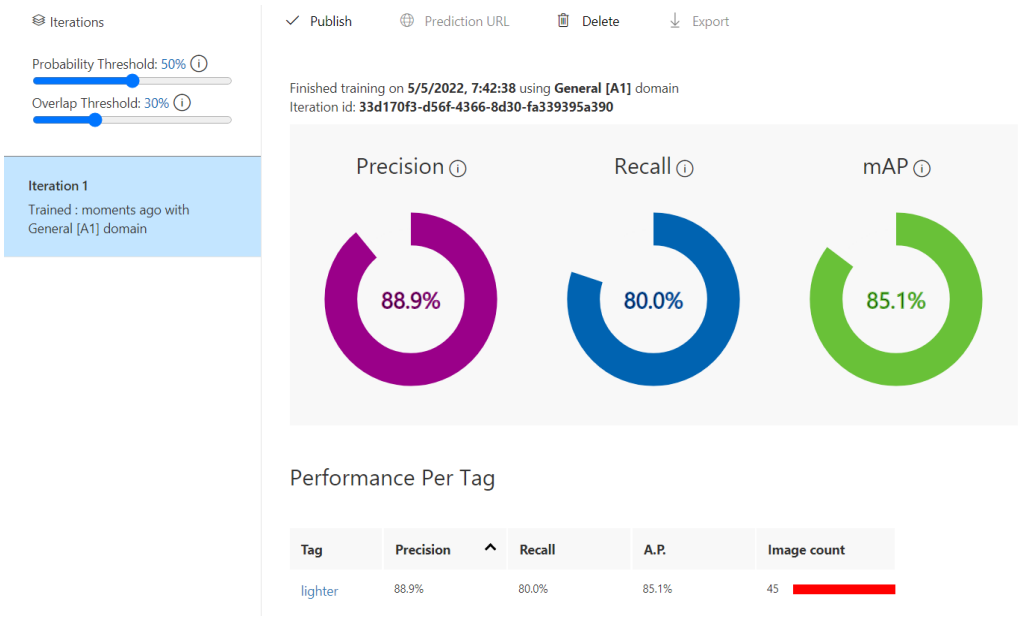

Fig. 13. Training finished and performance metrics (precision, recall and mean average precision)

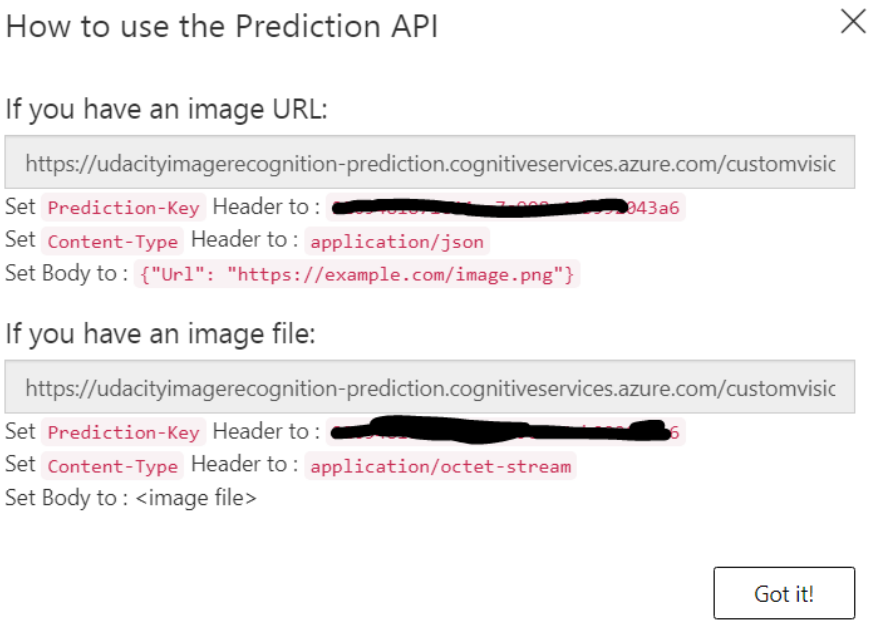

Fig. 14. Publishing the model to an endpoint

Snippet 7. Making a prediction and displaying a bounding box around the object.

```
def has_lighter_in_lugage(
    predictor: CustomVisionPredictionClient,
    image_path: Path,
    show_lighter_detection: bool = False,
    threshold: float = 0.95,
):
    results = predict(
        predictor, image_path, config.project_id,
config.publish_iteration_name
    )

    if show_lighter_detection:
        with open(image_path, "rb") as img_code:
            img_view_ready = Image.open(img_code)
            _, ax = plt.subplots()
            ax.imshow(img_view_ready)

    found_lighter = False
    for prediction in results.predictions:
        logger.debug(
            f"{prediction.tag_name}"
```

```
            ": {0:.2f}%".format(prediction.probability * 100)
        )
        found_lighter = prediction.probability > threshold or found_lighter
        if found_lighter and show_lighter_detection:
            bbox = prediction.bounding_box
            rect = patches.Rectangle(
                (
                    bbox.left * img_view_ready.width,
                    (1 - bbox.top) * img_view_ready.height,
                ),
                bbox.width * img_view_ready.width,
                -bbox.height * img_view_ready.height,
                linewidth=1,
                edgecolor="r",
                facecolor="none",
            )
            ax.add_patch(rect)
    return found_lighter
```

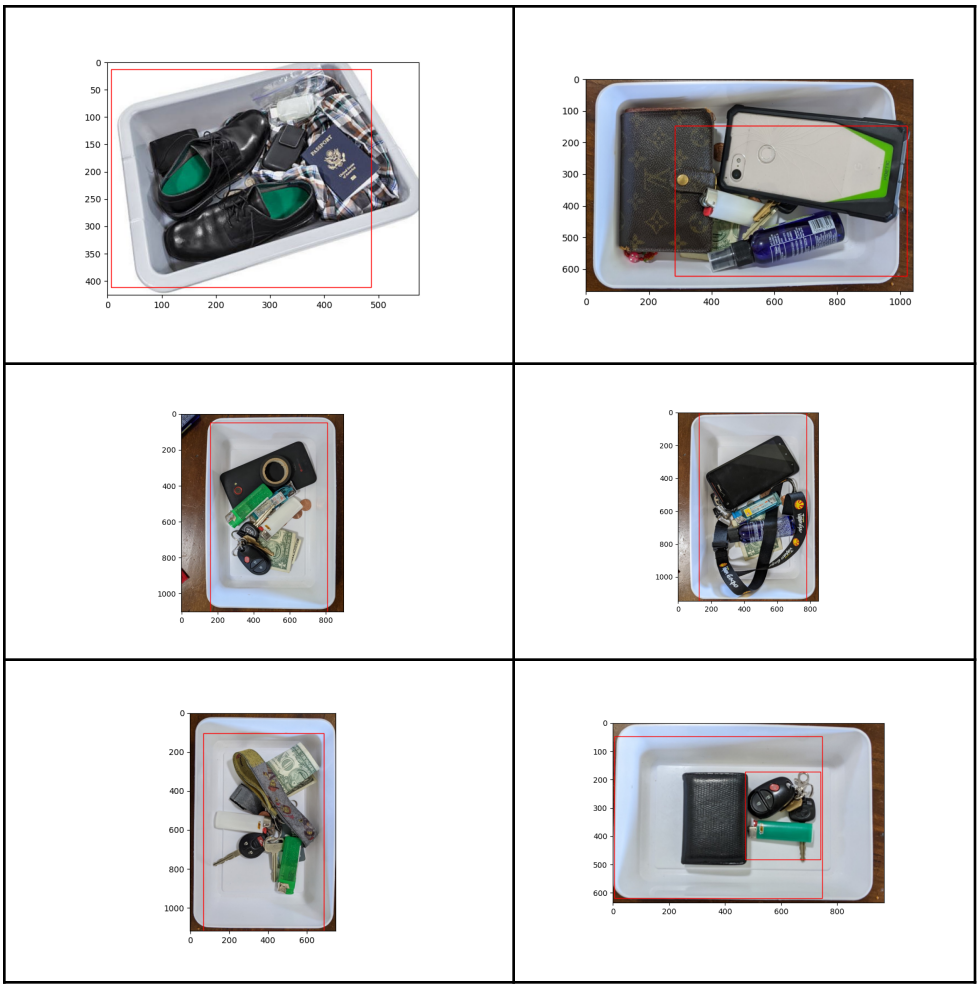

Fig. 15. Bounding boxes showing predictions on test images.

Validation and Metrics Monitoring

Merge various extracted information together for data validation and build the final solution

The driver program is too big to put it here, but it can be checked on the repository: [main.py](). As I am not willing to share my credentials due to security purposes, I will just show the output of running the program with no debug logs. It can already be verified that the data matches the ones of the ID, and for display purposes, I set manually the first passenger LuggageValidation to TRUE. Apart from the passenger Diogo (based on myself), no other passenger could have the identity verified, as they are compared to the person group built with a video of myself.

Snippet 8. Complete execution of the Automated Passenger Board Kiosk.

```
> poetry run python main.py
Getting video indexer access token...
Access Token:
eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTI
1NiIsInR5cCI6IkpXVCJ9.eyJBY2NvdW50SWQiOiJjZTVjMTAxNi0yOGI3LTQ0ZTItOWNjMi0zN
zBlNmMzZDNiNzYiLCJQZXJtaXNzaW9uIjoiQ29udHJpYnV0b3IiLCJFeHRlcm5hbFVzZXJJZCI6
IjEwNTMzNDI3MjY3OTA1ODYxODAzMCIsIlVzZXJUeXBlIjoiR29vZ2xlIiwiSXNzdWVyTG9jYXR
pb24iOiJUcmlhbCIsIm5iZiI6MTY1MTkwMTk4NywiZXhwIjoxNjUxOTA1ODg3LCJpc3MiOiJodH
RwczovL2FwaS52aWRlb2luZGV4ZXIuYWkvIiwiYXVkIjoiaHR0cHM6Ly9hcGkudmlkZW9pbmRle
GVyLmFpLyJ9.9veJmMlEymsoq9_TG0axuiHrAcHhYFi6MsMaciG5uS0
Getting video info for: 89fe0aeb73
Getting thumbnail from video: 89fe0aeb73, thumbnail:
556f540d-5c5a-48cd-b58a-eb1711fed033
Getting thumbnail from video: 89fe0aeb73, thumbnail:
9f16072f-8df3-4c2b-a571-2d8fa5b7a211
Getting thumbnail from video: 89fe0aeb73, thumbnail:
3033f152-e377-49f9-ba37-3988b37e8f96
Getting thumbnail from video: 89fe0aeb73, thumbnail:
cf19fb63-753c-44c0-9cb6-481d225fb789
Getting thumbnail from video: 89fe0aeb73, thumbnail:
6f05cbaa-46f4-4008-8b2d-587556abcd56
Getting thumbnail from video: 89fe0aeb73, thumbnail:
ac0c3c59-585e-4d65-9b19-094be9059abd
Getting thumbnail from video: 89fe0aeb73, thumbnail:
9e7063bf-cea0-4d80-8e1e-7ba0b57d17e3
Getting thumbnail from video: 89fe0aeb73, thumbnail:
f89a1840-da40-46cc-b2ee-6d22487d1af6
Dear Mr. Diogo Andrés Puelma,
You are welcome to the flight # LA-21 leaving at 11:00 from Santiago to
Concepción.
Your seat number is 3A, and it is confirmed
We did not find a prohibited item (lighter) in your carry-on baggage.
Thanks for following the procedure.
Your identity is verified so please board the plane.
Dear Mr. Manuel Ignacio Gaete,
You are welcome to the flight # SKY70 leaving at 14:00 from Concepción to
Vallenar.
Your seat number is 20C, and it is confirmed
We have found a prohibited item in your carry-on baggage, and it is flagged
for removal.
```

```
Your identity could not be verified. Please see a customer service
representative.
Dear Mr. Clynton Augusto Tomacheski,
You are welcome to the flight # Jet3 leaving at 8:00 from Temuco to Puerto
Montt.
Your seat number is 35B, and it is confirmed
We have found a prohibited item in your carry-on baggage, and it is flagged
for removal.

Your identity could not be verified. Please see a customer service
representative.
Dear Ms. Norma Alejandra Vásquez,
You are welcome to the flight # LA-21 leaving at 8:00 from Concepción to
Santiago.
Your seat number is 10F, and it is confirmed
We have found a prohibited item in your carry-on baggage, and it is flagged
for removal.

Your identity could not be verified. Please see a customer service
representative.
Dear Ms. Javiera Alejandra Strogonoff,
You are welcome to the flight # LA-23 leaving at 17:00 from Arica to Punta
Arenas.
Your seat number is 20E, and it is confirmed
We did not find a prohibited item (lighter) in your carry-on baggage.
Thanks for following the procedure.
Your identity could not be verified. Please see a customer service
representative.
```

No worries about the token info, as it is already expired, and I expect randomness on the signature portion of the JWT.

Snippet 9. Checking the logged token is expired.

```
> echo -n
eyJBY2NvdW50SWQiOiJjZTVjMTAxNi0yOGI3LTQ0ZTItOWNjMi0zNzBlNmMzZDNiNzYiLCJQZXJ
taXNzaW9uIjoiQ29udHJpYnV0b3IiLCJFeHRlcm5hbFVzZXJJZCI6IjEwNTMzNDI3MjY3OTA1OD
YxODAzMCIsIlVzZXJUeXBlIjoiR29vZ2xlIiwiSXNzdWVyTG9jYXRpb24iOiJUcmlhbCIsIm5iZ
iI6MTY1MTkwMTk4NywiZXhwIjoxNjUxOTA1ODg3LCJpc3MiOiJodHRwczovL2FwaS52aWRlb2lu
ZGV4ZXIuYWkvIiwiYXVkIjoiaHR0cHM6Ly9hcGkudmlkZW9pbmRleGVyLmFpLyJ9 | base64
--decode
{"AccountId":"ce5c1016-28b7-44e2-9cc2-370e6c3d3b76","Permission":"Contribut
or","ExternalUserId":"105334272679058618030","UserType":"Google","IssuerLoc
ation":"Trial","nbf":1651901987,"exp":1651905887,"iss":"https://api.videoin
```

```
dexer.ai/","aud":"https://api.videoindexer.ai/"}
```

Updated Manifest Table

The updated manifest table is in my repo: [updated manifest table.]

Monitor performance and usage metrics of each cognitive service

In the fig. 16, fig. 17 and fig. 18, monitoring of the cognitive services is shown. In particular, performance monitoring is depicted:

- Latency
- Total Calls
- Total Successful Calls
- Total Error Calls

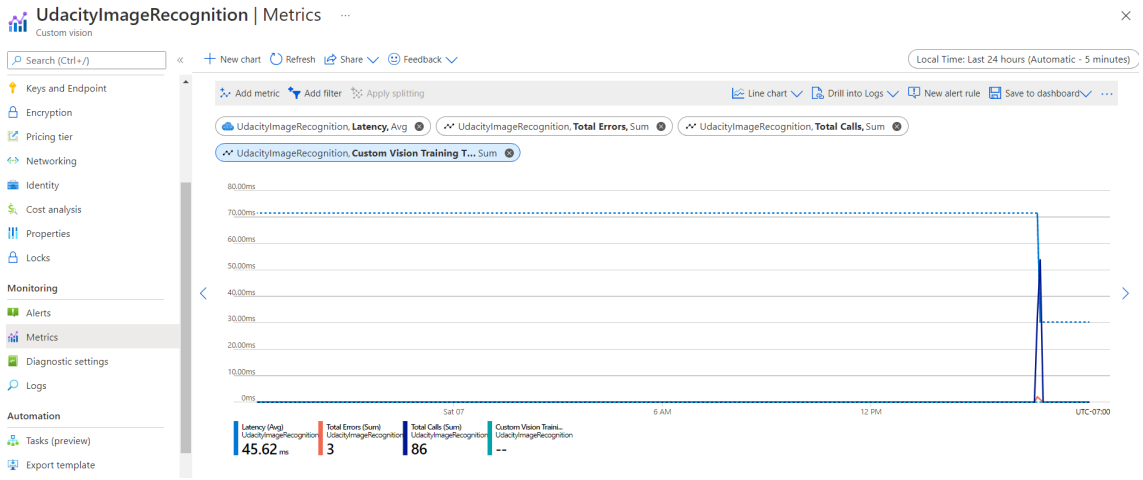

Fig. 16. Monitoring Custom Vision Service

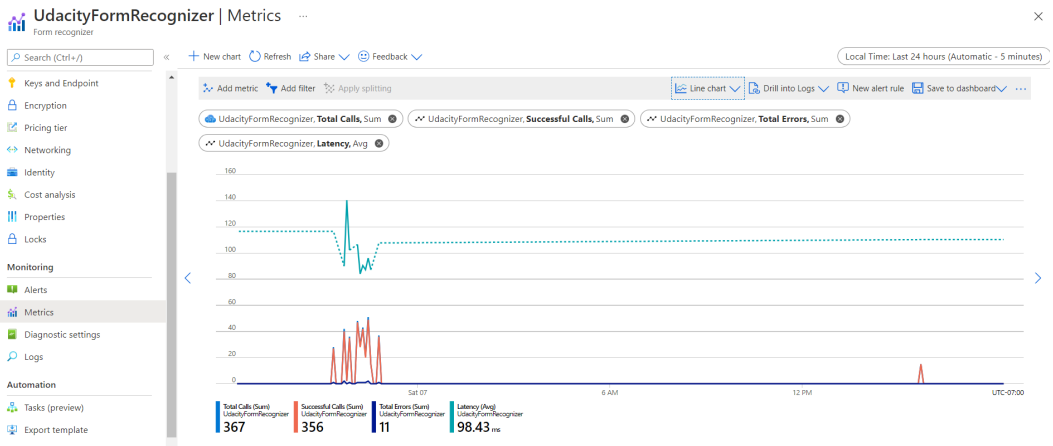

Fig. 17. Monitoring Form Recognizer Service

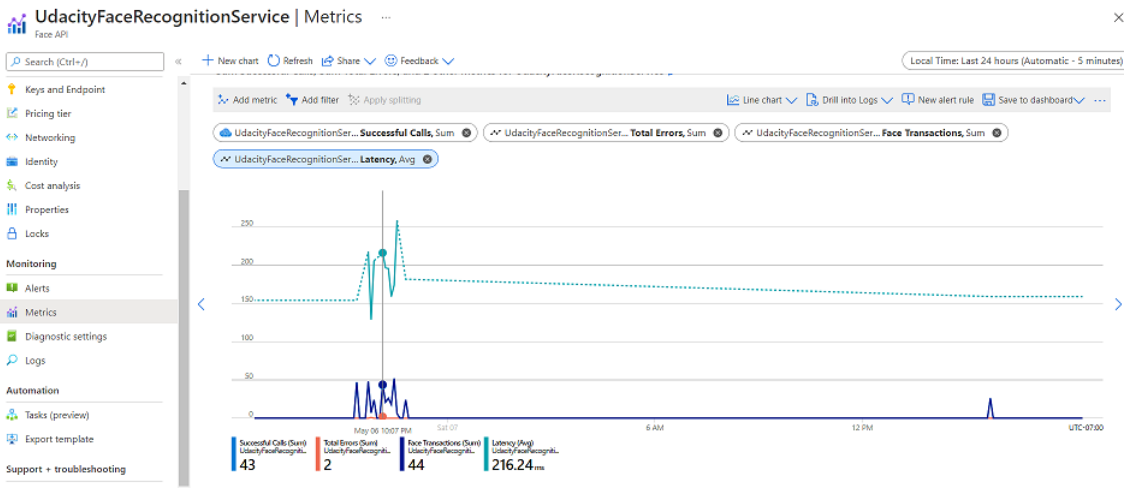

Fig. 18. Monitoring Face API service

Final Reflections

I ran into multiple issues, some annoying and some others very stressful. For example if I use blob storage and want to overwrite a file, I upload the file, fail on doing it, and then mark the overwrite box, it will not work.

The portal has several issues managing sessions and this caused me plenty of authentication and authorization issues that I took me time to debug. Moreover, some annoying features about video indexer, is that if I subscribe with an email but another type of account, I will get authentication and authorization errors and I didn't find a way to delete my account. The solution was creating a new account in the VdeoIndexer and using those credentials to continue my work.

Using the Face API and while comparing the frames of the video with my face in the ID, I got some interesting results. The fake passenger that is using my photo is Diogo, and I would expect that the similarity confidence was really high. In the snippet 10, I compared all the ID documents with the face of the video in which I wear glasses. Even though I got the highest confidence value, it is still low for instance if we consider a higher threshold (e.g. confidence above 50%):

Snippet 10. Compare documents ID with myself wearing glasses.

```
2022-05-05 06:12:47,391 video  INFO Comparing human-face4 to ca-dl-clynton
2022-05-05 06:12:47,391 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.18596.
2022-05-05 06:12:47,519 video  INFO Comparing human-face4 to ca-dl-diogo
2022-05-05 06:12:47,519 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.41758.
2022-05-05 06:12:47,626 video  INFO Comparing human-face4 to ca-dl-javiera
2022-05-05 06:12:47,626 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.09481.
2022-05-05 06:12:47,733 video  INFO Comparing human-face4 to ca-dl-manuel
```

```
2022-05-05 06:12:47,733 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.10889.
2022-05-05 06:12:47,841 video  INFO Comparing human-face4 to ca-dl-norma
2022-05-05 06:12:47,842 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.08895.
```

However, when I did the same but with my glasses off, I got a more sensible result:

```
2022-05-05 06:25:55,242 video  INFO Comparing human-face4 to ca-dl-clynton
2022-05-05 06:25:55,242 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.09248.
2022-05-05 06:25:55,347 video  INFO Comparing human-face4 to ca-dl-diogo
2022-05-05 06:25:55,347 video  INFO Faces are of the same (Positive)
person, similarity confidence: 0.67677.
2022-05-05 06:25:55,453 video  INFO Comparing human-face4 to ca-dl-javiera
2022-05-05 06:25:55,453 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.10802.
2022-05-05 06:25:55,555 video  INFO Comparing human-face4 to ca-dl-manuel
2022-05-05 06:25:55,555 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.20308.
2022-05-05 06:25:55,662 video  INFO Comparing human-face4 to ca-dl-norma
2022-05-05 06:25:55,663 video  INFO Faces are of different (Negative)
persons, similarity confidence: 0.09895.
```

Maybe light or quality of images (the quality of my webcam is not good) will impact results. So, how do we decide what threshold is appropriate?

WIth the object recognition model, per fig. 13, I got a supposedly good model. But when testing on an image with no lighter, it still tried to detect a lighter with high confidence (higher than 95%), as shown in fig. 19.

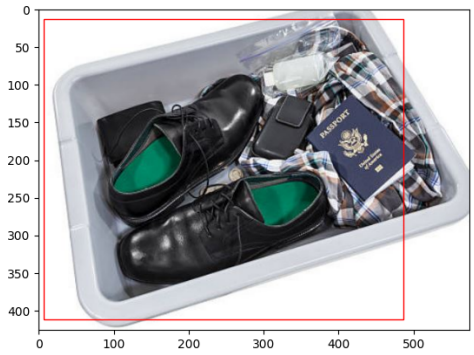

Fig. 19. False Positive on lighter detection.

This means we need to worry about having a high precision and very high recall, as missing a dangerous object (recall) will be dangerous for all the persons who will take the flight. Moreover,

having a low precision, will make the automation pointless, as it will require human confirmation to check if there was actually a lighter in this case.