Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Aplicação de Machine Learning em Quality Assurance

Relatório de Estágio

Sandra Maria Rassul Marques

Mestrado em Analítica e Inteligência Organizacional

Especialização em Quality Assurance Trainee

Lisboa | Novembro | 2022

Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Sandra Maria Rassul Marques

Aplicação de Machine Learning em Quality Assurance

Relatório de Estágio

Orientado por:

Sandra Maria Gonçalves Vilas Boas Jardim - Instituto Politécnico de Tomar

Miguel Pereira - Celfocus

Júri: **Presidente** Hélder Pestana, **Vogal** Vasco Silva e **Orientadora** Sandra Jardim - Instituto Politécnico de Tomar

Relatório de Estágio apresentado ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Analítica e Inteligência Organizacional

RESUMO

O presente relatório tem como objetivo sintetizar e expor as atividades, experiências e o conhecimento adquirido ao longo do estágio curricular. Pondo em prática todos os conhecimentos obtidos, no âmbito da unidade curricular de Estágio, pertencente ao Mestrado em Analítica e Inteligência Organizacional – Especialização em pertencente ao Instituto Politécnico de Tomar.

O estágio descrito teve a duração de 780 horas, tendo início no mês de Março de 2022 e duração de aproximadamente 5 meses. Durante este período participei no desenvolvimento do projeto Cognitive QA.

O estágio foi realizado na empresa tecnológica Celfocus, pertencente a Novabase, sediada no Parque das Nações na cidade de Lisboa. Esta empresa tem por missão o desenvolvimento de serviços e soluções inovadoras no setor das telecomunicações, o presente estágio decorreu em concreto no departamento de QA (Quality Assurance).

O estágio permitiu um contacto direto com a realidade de desenvolvimento tecnológico a nível empresarial, aquisição de novos conhecimentos e novas competências técnicas. No presente relatório expõe-se os conhecimentos adquiridos, as tarefas e os trabalhos desenvolvidos ao longo deste estágio, e com o aprofundamento da importância da área de Machine Learning para Quality Assurance.

Palavras-chave: Celfocus**,** Quality Assurance, Machine Learning.

ABSTRACT

This report aims to summarize and expose the activities, experiences and knowledge acquired during the curricular internship. Putting into practice all the knowledge obtained, within the scope of the Internship curricular unit, belonging to the Master in Analytics and Organizational Intelligence – Specialization in belonging to the Polytechnic Institute of Tomar.

The internship described, lasted 780 hours starting in March 2022, with the duration of approximately 5 months. During this period, I participated in the development of the project Cognitive QA.

The internship was conducted at the technological company Celfocus, belonging to Novabase, based in Parque das Nações in the city of Lisbon. This company aims to develop innovative services and solutions in the telecommunications sector, specifically in the QA (Quality Assurance) department.

The internship allowed direct contact with the reality of technological development at a business level, acquisition of new knowledge and new technical skills. In this report, the acquired knowledge, tasks, and work developed during this internship are exposed, and with the deepening of the importance of the Machine Learning area for Quality Assurance.

Keywords: Celfocus**,** Quality Assurance, Machine Learning.

AGRADECIMENTOS

Este relatório marca o fim de uma etapa muito importante e não ficaria completo sem o apoio e colaboração de várias pessoas. Quero assim expressar os meus maiores e sinceros agradecimentos a todos os que ajudaram a concretizá-lo.

Em primeiro lugar, gostaria de agradecer à Celfocus pela oportunidade de realizar o estágio curricular e ao Instituto Politécnico e Tomar pela consideração ao longo de todos os anos curriculares.

Um especial agradecimento ao meu supervisor da Celfocus, Miguel Pereira, pela sua simpatia, interesse no acompanhamento e desenvolvimento do estágio. Também a toda a minha equipa de Quality Assurance, pela ajuda prestada e sua disponibilidade, tempo e orientação ao longo deste percurso.

Gostaria de agradecer a minha orientadora e coordenadora do Mestrado, Sandra Jardim, pela sua disponibilidade e apoio na concretização deste projeto.

Os meus agradecimentos finais vão para a minha família e amigos pelo apoio, o tempo, pelos incentivos ao longo da realização deste trabalho e, sobretudo, pelo encorajamento dado.

A todos, muito obrigada!

GLOSSÁRIO

Termo	Descrição
Quality Assurance	Ramo da informática que visa a garantir a qualidade do desenvolvimento de um produto ou serviço, através do cumprimento de critérios e métodos.
Framework	Termo genérico para uma estrutura sobre a qual podem ser construídas ferramentas.
App (Aplicação)	Programa de software presente em telemóveis, computadores e outros dispositivos inteligentes que apresentam diversas funções de acordo com a sua finalidade.
Dataset	Conjunto de dados, representados em tabelas utilizados em processos de análise de dados.
Test Cases	Série de ações executadas de modo a testar o equipamento ou software, normalmente executado pela equipa de *Quality Assurance*.
Inteligência Artificial	Capacidade de uma máquina reproduzir competências similares às competências humanas, tais como aprendizagem raciocínio e tomada de decisões.
Open source	Denomina um tipo de software, cujo código fonte é aberto para qualquer individuo, ou seja, de uso livre.
Machine Learning	Área da Inteligência Artificial, tem como objetivo facilitar e melhorar a previsão de resultados de softwares, através de algoritmos e modelos.

TABELA DE ACRÓNIMOS

Acrónimo	Descrição
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CSV	Comma-separated values
GUI	Graphic User Interface
HLD	High Level Definition
UI	User Interface
ML	Machine Learning
QA	Quality Assurance

Índice

Índice de Figuras

Capítulo 1 Introdução

Neste primeiro capítulo é feito um enquadramento do projeto desenvolvido ao longo do estágio, assim como o seu âmbito. Seguidamente, é feita uma breve descrição da empresa referindo alguns aspetos importantes a nível tecnológico, terminando o capítulo com uma breve descrição da estrutura do documento apresentado.

1.1 Enquadramento e Objetivos

No âmbito da unidade curricular de Estágio, do 2º ano do curso de Mestrado em Analítica e Inteligência Organizacional – Especialização em *Quality Assurance*, foi realizado um estágio curricular, com duração de 750 horas, correspondente a 5 meses, na empresa Celfocus.

O projeto que teve intervenção direta no âmbito do trabalho de estágio foi o projeto Cognitive QA, uma solução tecnológica desenvolvida para o setor de *Quality Assurance* de forma a assegurar e melhorar, a qualidade dos testes realizados pela empresa.

Os objetivos estabelecidos para a realização deste estágio, centram-se na aquisição de conhecimentos, ganho de experiência e no desenvolvimento de competências na área de estudo.

1.2 Descrição da empresa Celfocus

A entidade de acolhimento do estágio Celfocus[1], fundada no ano de 2000, a Celfocus é uma empresa tecnológica do Grupo Novabase que fornece serviços de integração de sistemas, utilizando tecnologias de ponta, com aposta nos segmentos de Digital e Cognitive.
Visa impulsionar a inovação de Produtos e Serviços, promovendo componentes digitais inovadoras e trazendo valor aos desafios mais complexos e críticos de negócio. Desde a Estratégia até às Operações,

[1] Disponível em Home :: celfocus

a Celfocus colabora em áreas como Inteligência Artificial, Cognitive Automation e Digital, com benefícios transversais ao Negócio e à Tecnologia.

O Digital – num contexto de crescimento e aumento significativo da complexidade de dados – fomenta a inovação e a disrupção em larga escala. Assim, as principais ofertas da Celfocus estão centradas em: Transformação Digital, Cognitive e Automação de Redes, com foco no desempenho de negócio e na experiência do consumidor.

A metodologia Agile da Celfocus está alinhada com as boas práticas da indústria, assente na redução de risco e dos períodos de comercialização. Logo, os clientes têm uma visão holística do que traz, ou não, valor, com base numa arquitetura centrada no consumidor, acompanhada por uma abordagem analítica e na reutilização de componentes.[2]

a. Localização

A empresa possui instalações espalhadas pelo mundo, apresentando nesta fase, um maior foco de recursos em Portugal, com escritórios localizados em Lisboa e no Porto, tendo clientes a nível mundial em mais de 25 países, a Celfocus ajuda organizações a transformar o seu negócio, melhorando a sua vantagem competitiva e o seu desempenho. [Figura 1- Localizações dos escritórios da Celfocus][3]

O presente estágio decorreu maioritariamente em regime remoto, contudo mensalmente foram efetuadas reuniões presencias foram realizadas nos escritórios de Lisboa.

Figura 1- Localizações dos escritórios da Celfocus

[2] Imagem obtida em (4) Celfocus: Overview | LinkedIn
[3] Imagem obtida em Who We Are :: celfocus

1.3 Estrutura do documento

O presente documento, foi realizado tendo por suporte as normas base para a elaboração de relatório de estágio de Mestrado da Instituição. Encontra-se organizado de forma a permitir que o leitor compreenda a evolução e o progresso de todo o estágio.

O relatório encontra-se dividido em três partes, com um total de 6 capítulos.

A primeira parte, ou parte introdutória, visa informar o leitor acerca das motivações, informações acerca da empresa e metodologias utilizadas pela mesma para o desenvolvimento do projeto, incluído assim o Capítulo 1 (Introdução) e o Capítulo 2 (Enquadramento do projeto).

A segunda parte, refere o âmbito do estágio que apresenta uma breve sumarização dos conceitos utilizados no projeto, bem como o seu desenvolvimento e a explicação do desenvolvimento de certas componentes ao longo do estágio, incluindo assim o Capítulo 3 (Projeto Cognitive QA) e o Capítulo 4 (Componentes Desenvolvidas).

Por último, a terceira parte, apresenta as considerações finais do projeto, sendo feita uma análise da importância o uso de ferramenta tecnológicas como *machine learning* para a área em estudo no Capítulo 5 (Conclusão).

Capítulo 2 Metodologias e Planeamento

O seguinte capítulo pretende descrever as metodologias utilizadas na empresa a nível de trabalho, assim como a metodologia adotada pela equipa de desenvolvimento do projeto. Em adição é feita uma descrição do planeamento do projeto e uma breve sumarização das bibliotecas e ferramentas utilizadas para o desenvolvimento do mesmo.

2.1 Metodologia Utilizada

É frequente a utilização de uma metodologia de desenvolvimento nas empresas, com o intuito de diminuir falhas, atrasos e custos na produção. A Celfocus utiliza as metodologias *Waterfall* e *Agile*, tendo em conta produto requerido pelo cliente. Para a produção de produtos, projetos e serviços a metodologia mais utilizada é *Agile*, enquanto para projetos onde a vertente técnica é predominante é utilizada a metodologia *Waterfall*.

A metodologia *Waterfall*, de significado "Cascata", apresenta um processo sequencial, com base na figura 3 consegue-se verificar que esta metodologia expõe apenas um ciclo. Ao contrário da *Agile*, que apresenta vários ciclos, eliminando algumas das limitações da metodologia mencionada acima, obtendo assim diferentes resultados e uma maior margem para modificações. [Figura 2-Comparação do funcionamento das metodologias Waterfall e Agile][4]

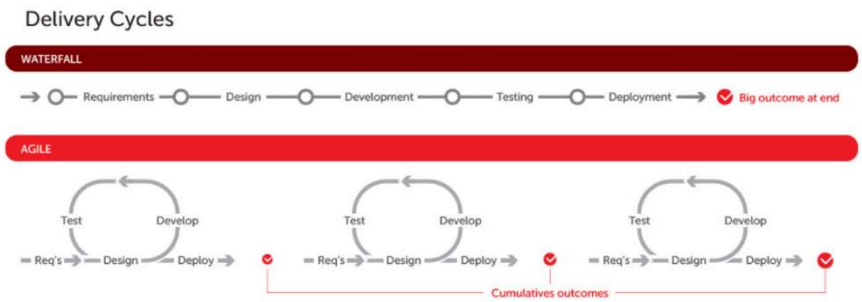


Figura 2-Comparação do funcionamento das metodologias Waterfall e Agile

[4] Imagen obtida em No formulas :: Call - Celfocus News for All

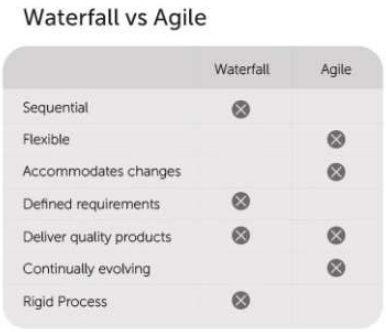


Figura 3 – Vantagens e desvantagens entre os métodos Waterfall e Agile

Para o projeto realizado utilizou-se a metodologia *Agile*, esta foca-se num conjunto de técnicas e práticas para ajudar na eficiência, rapidez e flexibilidade da gestão de projetos, permitindo assim uma evolução constante nas equipas e entrega rápida do projeto. [Figura 3 – Vantagens e desvantagens entre os métodos Waterfall e Agile][5]

Com o uso desta metodologia os processos tornam-se mais simples, dinâmicos e iterativos até ao produto final. [Figura 4- Funcionamento da Metodologia Agile][6]

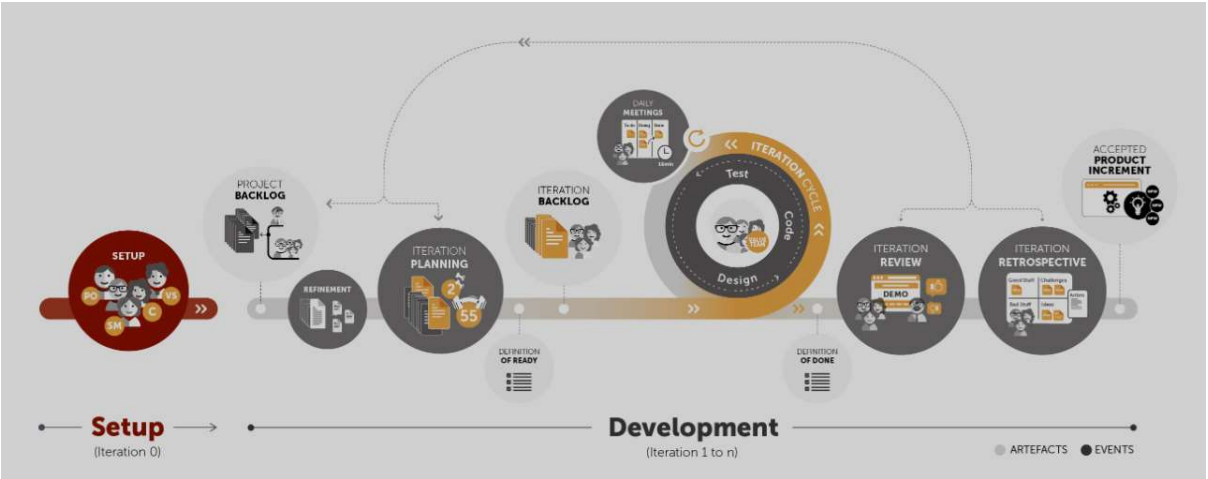


Figura 4- Funcionamento da Metodologia Agile

[5] Imagen obtida em No formulas :: Call - Celfocus News for All
[6] Imagen obtida em Make it Happen :: celfocus

2.2 Equipa - Quality Assurance

A equipa *de Quality Assurance* tem como objetivo garantir excelência do produto, atendendo às necessidades, requisitos e expectativas do cliente, através do fornecimento de soluções robustas e de alta qualidade, havendo um controlo na qualidade do produto a entregar, evitando assim defeitos ou possíveis problemas.

De acordo com a empresa a garantia de excelência de produto[7] é possível através das seguintes componentes:

- **Qualidade da Solução e Satisfação do Cliente:** Agindo como uma equipa independente e imparcial. A atitude encontra-se focada na garantia da qualidade das soluções dos clientes.
- **Esforços de teste otimizados:** Com base na experiência na indústria Telco e metodologias de testes, é possível fornecer um forte conhecimento de onde e o que testar, de modo a maximizar o retorno do esforço.
- **Configuração rápida do projeto e entrega oportuna:** Os clientes aproveitam o conhecimento, experiência e ativos de teste comprovados.

2.3 Planeamento do projeto

O presente estágio teve início a 5 de março de 2022 e concluído a 22 de julho de 2022. As primeiras semanas consistiram na integração e adaptação a empresa, com a finalidade de integração na equipa e no projeto.

O projeto já tinha dado início com a recolha de pré-requisitos, através de entrevistas com os colaboradores da Celfocus. Com base nos resultados obtidos, foi construído um modelo e realizada uma análise. Através dessa análise, deu-se início ao desenvolvimento da ferramenta Cognitive QA.

Foi realizado um estudo de todas as tecnologias e softwares necessários e ferramentas já implementadas ou por desenvolver, permitindo assim uma maior perceção acerca do projeto. Paralelamente, não só foram realizadas formações no âmbito da empresa, bem como para o projeto,

[7] Disponível em It's Quality Time! :: Call - Celfocus News for All

de forma a consolidar os conhecimentos na linguagem Python.

2.4 Ferramentas utilizadas

O desenvolvimento do projeto Cognitve QA, já havia iniciado, fazendo com que algumas das ferramentas e tecnologias a utilizar já se encontrassem definidas. Sendo necessário uma análise prévia das bibliotecas utilizadas, de forma a conhecer o projeto para dar continuidade ao desenvolvimento do mesmo.

Este projeto baseou-se, maioritariamente, em desenvolvimento na linguagem Python. Contudo para a evolução do mesmo foram necessárias algumas ferramentas tecnológicas. Ferramentas a nível de ambiente de trabalho, para a comunicação entre a equipa e para o desenvolvimento do código necessário. Também foram necessárias bibliotecas de programação e pacotes que ajudaram a complementar as tarefas e os desafios propostos.

Uma grande vantagem, destas ferramentas e bibliotecas, é o facto de serem ferramentas *open source.* Existindo uma vasta comunidade a nível global que utilizam estas ferramentas diariamente, havendo um enorme nível de suporte e documentação técnica.

Em seguida, estão descritas as tecnologias, linguagem e bibliotecas utilizadas ao longo do projeto.

a. Tecnologias

Microsoft Teams

O Microsoft Teams[8] foi a aplicação de comunicação utilizada ao longo de todo o estágio entre a equipa.

Permitindo às equipas a possibilidade de realizarem chamadas, videoconferências e facilitando na colaboração no desenvolvimento de documentos e outros processos do Office365.

Facilitando assim a comunicação e organização das equipas em modo trabalho híbrido.

[8] Disponível em Video Conferencing, Meetings, Calling | Microsoft Teams

Visual Studio Code

O Visual Studio Code[9] é um editor de código, utilizado durante o desenvolvimento do projeto.

Desenvolvido pela Microsoft, é um editor de código *open source*, apresenta enumeras funcionalidades, linguagens de programação fornecendo uma coletânea de extensões e de bibliotecas.

Google Colaboratory

O Google Colaboratory[10], conhecido por Colab, foi utilizado, para a realização de testes e de tratamento de dados. É um editor de código focado apenas na linguagem Python que permite realizar, de forma simples e eficaz o estudo de áreas como *Machine Learning*, *Data Analysis e Artificial Intelligence*. Este editor funciona como um Notebook apresentando assim o código fonte e o respetivo resultado.

Este editor em conjunto com o Visual Studio Code facilitaram o desenvolvimento do projeto.

Git

O Git[11] é um sistema de controlo de versão, open source, desenvolvido pelo criador do kernel Linux[12], na qual permitiu a monotorização do desenvolvimento do projeto e no controlo de versões.

Esta ferramenta permite ver o código de outros colaboradores, as modificações feitas e os históricos. Possibilitando assim um maior controlo sobre os projetos e os colaboradores, diminuído assim possíveis falhas.

[9] Disponível em Visual Studio Code - Code Editing. Redefined
[10] Disponível em Google Colab
[11] Disponível em Git (git-scm.com)
[12] Linus Torvalds, criador do sistema operativo Linux.

b. Linguagem

Python

Python[13], é uma linguagem de programação de alto nível, dinâmica, orientada a objetos e utilizada em diversas plataformas. Sendo uma das linguagens mais adequadas para o desenvolvimento deste projeto. Apresenta uma sintaxe simples e de fácil compreensão com um grande número de bibliotecas nativas e de terceiros, é bastante utilizada nas áreas de inteligência artificial, *machine learning* e desenvolvimento web.

c. Bibliotecas de programação

As bibliotecas utilizadas para o tratamento de dados e para o desenvolvimento da programação Python foram as seguintes:

- **Docx2python**

  Biblioteca Python usada para interação com documentos (ficheiros .docx). Tendo possibilidade de configurar cabeçalhos, rodapés, notas de rodapé de texto, notas finais, propriedades do documento e imagens.

- **Openpyxl**

  Biblioteca Python usada para a leitura e edição de ficheiros Excel (xls/xlsm/xltx/xltx/ xltm).

- **Python-docx**

  Biblioteca Python usada para a criação e atualização de documentos (ficheiros .docx).

- **Pandas**

  Pacote Python utilizado, principalmente, para a análise de dados. Fornecendo uma forma rápida, flexível e intuitiva para analisar os dados.

[13] Disponível em Welcome to Python.org

- **Numpy**

  Proporciona uma representação estética e agradável das estruturas de dados, facilitando a leitura aos utilizadores.

Relativamente às tarefas designadas para o desenvolvimento de algoritmos de *Machine Learning* foram utilizadas as seguintes bibliotecas:

- **Pytesseract**[14]

  Através do uso do OCR (optical character recognition) esta biblioteca tem como capacidade ler e devolver o texto embebido nas imagens. O uso desta ferramenta foi fundamental para a realização de certas tarefas, relativas às áreas de *Deep Vision* e *Computer Vision*.

- **Open cv (Open-Source Computer Vision)**[15]

  Biblioteca designada para o uso de *Computer Vision e Machine Learning*, para o processamento de imagem e realização de tarefas. Esta ferramenta é utilizada no reconhecimento facial e objetos.

  Esta biblioteca foi fundamental para o projeto, principalmente, nas tarefas de reconhecimento de objetos e leitura de texto nas imagens.

- **Tkinter**[16]

  Proporciona uma representação gráfica e criação de aplicações GUI (*Graphic User Interface*). Esta biblioteca foi um grande auxílio na criação de tabelas do projeto.

Todas estas bibliotecas e bibliotecas adicionais, utilizadas ao longo do desenvolvimento do projeto, estão descritas detalhadamente no Capítulo 6 deste documento.

[14] Disponível em Read Text from Image with One Line of Python Code | by Dario Radečić | Towards Data Science
[15] Disponível em OpenCV: cv::MSER Class Reference
[16] Disponível em Python GUI Programming With Tkinter – Real Python

Capítulo 3 Projeto Cognitive QA

Para a evolução das componentes da plataforma e de modo a cumprir os objetivos propostos para o estágio, foi necessário um entendimento de um conjunto de técnicas e bibliotecas já adotados pela equipa de desenvolvimento.

Neste capítulo serão descritos os processos utilizados, os conceitos base para o desenvolvimento do projeto e uma explicação da importância do uso destes conceitos para a área de *Quality Assurance*.

3.1 Inteligência Artificial

A Inteligência Artificial foi um termo criado em 1965 por John McCarthy[17], quando organizava a conferência *Dartmouth Summer Research Project on Artificial Intelligence*, (McCarthy, Minsky, Rochester, & Shannon, 1995), definia-se como a capacidade de uma máquina conseguir reproduzir capacidades ou inteligência a nível humano, ou seja, capaz de realizar as tarefas e tomar decisões. (Andresen, 2002)

Através do surgimento da tecnologia e o aumento significativo de dados, foi possível armazenar e processar dados suficientes para que a inteligência artificial evoluísse de forma global e se tornasse uma ferramenta indispensável na atualidade. Diferentes sistemas de AI são responsáveis por diferentes objetivos como aprendizagem, tomada de decisão e deteção de objetos, subáreas relativas a Deep learning e Machine Learning. [Figura 5- Explicação das diferentes áreas de Inteligência Artificial] [18]

Esta área tem tido grande impacto em muitas áreas ao longo dos anos e tem vindo a apresentar um crescimento exponencial. A inteligência artificial está presente no dia a dia nas mais variadas formas, e por vezes sem ser percetível. Pequenas rotinas como uso de telemóveis e computadores, carros automáticos, são alguns dos exemplos da existência de inteligência artificial no quotidiano. (Ertel, 2017)

[17] Disponível em John McCarthy: father of AI | IEEE Journals & Magazine | IEEE Xplore
[18] Imagem disponível em Computer Vision .vs Machine Learning .vs Deep Learning | Guide to AI applications | ByteAnt

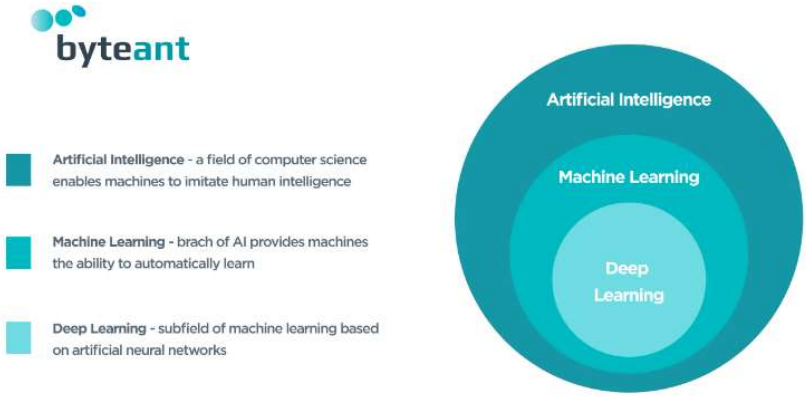


Figura 5- Explicação das diferentes áreas de Inteligência Artificial

a. Machine Learning

Machine learning é uma subárea da inteligência artificial, focada na análise de dados, treino e construção de modelos analíticos. Tem como objetivo a procura de algoritmos, encontrando padrões específicos para a previsão de resultados e na tomada de decisões. Através de dados e modelos de aprendizagem, é possível efetuar inúmeros cálculos de modo a produzir decisões e previsões com resultados fiáveis. (Alpaydin)

Os algoritmos de *Machine learning* dividem-se em três categorias[19]:

- Aprendizagem supervisionada, consiste numa aprendizagem prévia, na qual um conjunto de dados já se apresenta classificado, as entradas e as saídas são conhecidas e serão posteriormente utilizadas para outras instâncias.

- Aprendizagem não supervisionada, não consiste numa aprendizagem prévia, o intuito é relacionar os conjuntos de dados que não se encontram classificados, ou seja, identificar padrões de modo a classificá-los.

- Aprendizagem por reforço, é método contruído por base de experiência, aprende com os resultados e decide qual é a ação que será executada, concluindo o algoritmo vai escolher qual é o melhor caminho com base na experiência de tentativa erro.

Tal como a inteligência artificial é predominante no quotidiano, o mesmo acontece na área de *Machine*

[19] Disponível em Machine learning: o que é, para que serve + exemplos (zendesk.com.br)

learning, sendo esta uma forma rápida, eficaz e precisa na produção de resultados, de acordo com os seguintes exemplos:

- Deteção de fraudes, através do uso de algoritmos ML é possível detetar e combater ações fraudulentas, garantindo assim a segurança das empresas e dos clientes;
- Publicidades e recomendações online, com base na aprendizagem não supervisionada é possível agrupar campanhas de marketing para os clientes;
- Diagnostico médico, com o uso de *chatbots* [20]e softwares de reconhecimento facial é possível fazer um diagnóstico ao paciente através da informação recolhida ou pela identificação de padrões nas imagens.

i. Deep Learning

Deep Learning é uma evolução *de Machine Learning*, apresenta um conjunto de algoritmos que tem como objetivo mimetizar o cérebro humano. Consiste no uso de *Artificial Neural Network* (ANN), um conjunto de neurónios em camada que comunicam entre si, desde a entrada e saída de informação, processando assim os dados da mesma forma que o cérebro humano.

O funcionamento deste estudo é através do fornecimento de vários exemplos rotulados (com uma descrição do exemplo) para um banco de dados específico, o *Deep Learning* é capaz de extrair os padrões comuns existentes entre os exemplos, e de seguida transforma-los numa equação matemática de modo ajudar a classificar informações futuras

É um método eficaz, simples e fácil de desenvolver e implementar, esta técnica pode ser utilizada em reconhecimento de fala, deteção de objetos, classificação de imagens, entre outros. Com o objetivo de concretizar sistemas dinâmicos e preditivos que estão em constate melhoramento. (Yan, 2016)

[20] Software que simula conversas com utilizadores através de mensagens. Diponível em ChatBot | AI Chat Bot Software for Your Website

b. Computer Vision

Computer vision é outra área da inteligência artificial, que permite aos sistemas informáticos devolverem informações essências relativas a imagens, vídeos e outros informações visuais, auxiliando na tomada de decisão e na recomendação de acordo com a informação obtida. [

Figura 6 - Exemplo de Computer vision, através do uso do algoritmo YOLO com base na deteção e classificação de objetos] [21]

De certa forma funciona como a visão humana, apenas é possível detetar ou prever objetos através da observação e análise dos mesmos. Para isto se concretizar é necessário um banco de imagens, com as suas devidas designações e por fim sujeitá-las a técnicas de deteção de objetos.

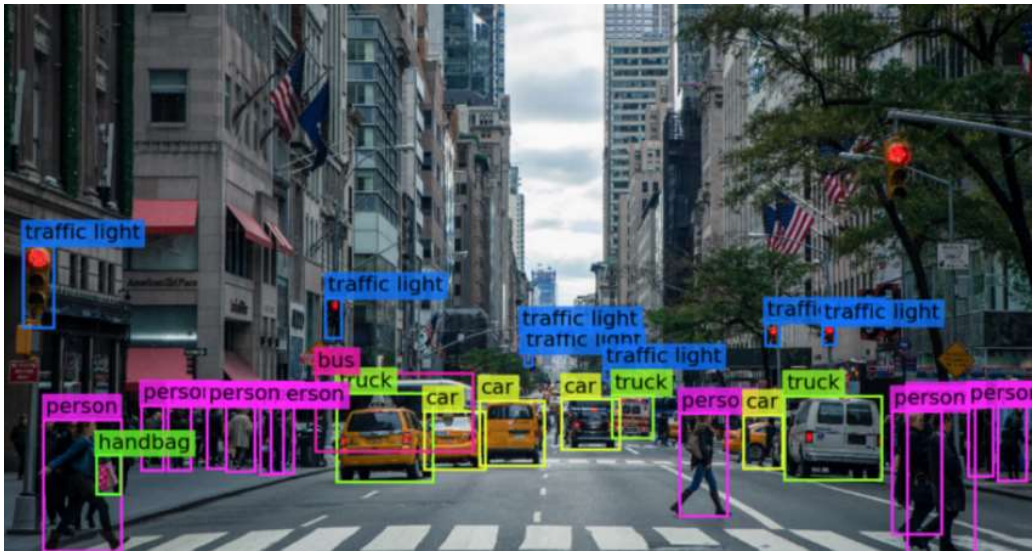


Figura 6 - Exemplo de Computer vision, através do uso do algoritmo YOLO com base na deteção e classificação de objetos

Esta técnica já se encontra integrada em alguns conceitos do nosso quotidiano como:

- Veículos autónomos, permite com que os veículos tenham uma maior perceção do ambiente a seu redor. As câmaras embutidas capturam imagens e vídeos de diferentes ângulos e através do uso de Computer Vision é possível identificar os objetos ao seu redor e possíveis perigos encontrados.

[21] Imagem obtida em You are being redirected... (analyticsinsight.net)

- Reconhecimento facial, através de algoritmos de computer vision é possível detetar características faciais em imagens, ajudando assim na identificação de perfis.
- Saúde, este conceito tem sido vastamente utilizado nos avanços da tecnologia na saúde, estes algoritmos ajudam a automatizar tarefas e na deteção de doenças através de imagens de pele ou raio x.

3.2 Cognitive QA

O projeto Cognitive QA teve origem devido a necessidade de otimização dos processos e ferramentas de *Quality Assurance* utilizados nos futuros projetos com clientes.

Foi necessário a aplicação de técnicas de AI e *Data Science* em processos e dados de testes de software. Para atingir os objetivos foi fundamental a experiência dos colaboradores da equipa QA da Celfocus e os respetivos dados obtidos de estimativa e execução dos projetos passados. Com o objetivo de prever e projetar os futuros projetos foi necessário obter conhecimentos e dados sobre projetos passados.

A ferramenta desenvolvida tem como objetivo a analise de documentos HLD (*High Level Definition*)[22] específicos da Celfocus, de modo a facilitar e simplificar a leitura e análise do mesmo. Retornando apenas as informações relevantes do projeto descrito no documento.

Através do histórico obtido por outros projetos, a ferramenta Cognitive QA, é capaz de realizar uma predição no número de *test cases* [23]que um projeto poderá gerar.

A existência de plataformas de automação da execução de testes já é comum, contudo não existem ferramentas que permitam a produção automatizada de planos de testes, nem a estimativa do esforço inerente à sua execução ou até do volume de bugs (defeitos ou não conformidades) que é expectável existirem e serem identificados em cada projeto.

Este projeto, apresenta um conjunto de ferramentas que, quando utilizadas pelos engenheiros de QA, lhes permitirão ser mais produtivos e precisos nas estimativas de qualidade e do esforço para validação

[22] Documentos com informações relativas a projetos da Celfocus.
[23] Conjunto de ações ou instruções que visam a validar as funcionalidades de um produto ou projeto. Disponível em What Is a Test Case? Examples, Types and Format - Applause

da qualidade do software entregue aos clientes.

3.3 Aplicações de Inteligência Artificial em Quality Assurance

Num mundo em que tecnologias como a Inteligência Artificial, estão a revolucionar a forma como os serviços são prestados e usufruídos pelos clientes, é surpreendente que essas tecnologias sejam ainda utilizadas de forma bastante incipiente nalguns dos passos do processo da tecnologia e nas empresas.

É abundante a existência de plataformas de automação de execução de testes, porém nem todas as ferramentas possuem a produção automatizada, de forma a inovar as ferramentas utilizadas pelos colaboradores e simplificar o processo, um dos objetivos da empresa foi incorporar estes mecanismos nesta nova ferramenta.

Com a utilização da Inteligência Artificial e *Machine Learning* é possível melhorar a rapidez e eficiência com que os testes são executados, diminuído também a margem de erro, evitando os lapsos gerados por fatores humanos. Esta ferramenta proporciona uma forma inteligente de gestão e projeção de futuros projetos, com o uso destas tecnologias inovadoras é possível realizar previsões cada vez mais precisas e facilitando também a analise de documentos.

Capítulo 4 Componentes Desenvolvidas

O presente capítulo descreve de todas as componentes desenvolvidas pela equipa durante a duração do estágio. É apresentado uma listagem por ordem cronológica as tarefas propostas com uma breve descrição do problema, seguida da solução adotada e os resultados obtidos.

4.1 Tarefas iniciais (pré-requisitos e bibliotecas)

Tendo em consideração que o projeto já tinha iniciado, foi necessário um estudo e uma análise prévia da documentação necessária e realização um ficheiro de configuração, incluindo todas as bibliotecas utilizadas (ficheiro txt, *requirements*).

Com a realização do ficheiro, mencionado acima, o utilizador passa a ter a possibilidade de instalar automaticamente todas as bibliotecas utilizadas de forma rápida e eficiente. O ficheiro de configurações, encontra-se no Capítulo 6 dos anexos e representa uma documentação do projeto, apresentando assim o nome de cada biblioteca utilizada seguida de uma breve descrição.

4.2 Soma dos valores da coluna Test Cases

O objetivo desta tarefa consiste em apurar estimativas do número de *test cases* para cada uma das atividades. Para este desenvolvimento foi utilizada a linguagem Python, teve-se em consideração a estrutura já previamente utilizada, da biblioteca Tkinter (explicação no Capítulo 2 - 2.1c).

Tendo em consideração que para cada *businesses scenarios*[24] estão associados *test cases*, procedeu-se a incrementação dos mesmos. Retornando o somatório de todos os *test cases* para todos *businesses scenarios* (comando: `total_tc += bsd[“tot_tc])` do documento apresentado. [Figura 7 - Criação da variável total test cases]

[24] É uma descrição completa do problema de negócios, quanto aos seu termos, arquitetura entre outros requisitos

```
if bsd:    # if there are business scenario details
    rid=rid+1
    lid=0
    total_tc += bsd["tot_tc"]  # add to test case count
```

Figura 7 - Criação da variável total test cases

Uma vez que a estrutura apresentada da tabela, era em árvore, foi acrescentado uma raiz através do comando `tree_add_root`(), de modo a apresentar os valores totais na tabela, acrescentado assim uma linha no final da tabela com o resultado calculado anteriormente da variável `total_tc` [Figura 8- Adição do Total de test cases a tabela].

```
# add a row with the total count of test cases
tree_add_root(tc_tree, rid+1, "{:>100}Total Test Cases".format(""), (total_tc,"","","",""))
```

Figura 8- Adição do Total de test cases a tabela

Por fim é apresentada a tabela com as atividades (primeira coluna da tabela), os testes cases (segunda coluna da tabela) realizados, o seu tipo (terceira coluna da tabela) e a sua complexidade (quarta coluna da tabela) com a adição da nova linha de total de test cases. [Figura 9 - Tabela Test Cases, ferramenta Cognitive QA]

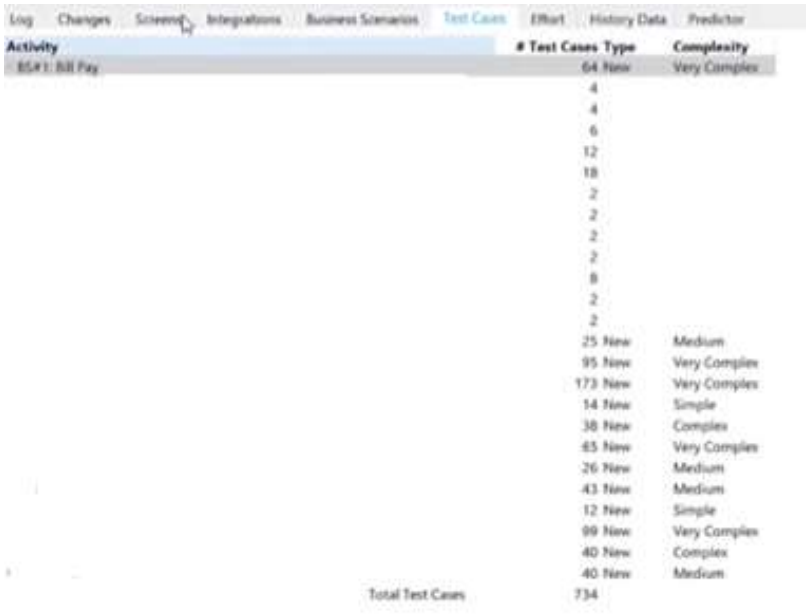


Figura 9 - Tabela Test Cases, ferramenta Cognitive QA

Nota: Todas as informações consideradas confidencias, foram ocultadas da tabela em questão.

4.3 Criação da Tabela History Data

Para a realização desta tarefa foi utilizada a biblioteca Tkinter, de forma a manter a estrutura da tabela. Deu-se início com uma análise dos dados e transformação dos mesmos, de modo a serem carregados posteriormente. Como tal criou-se a função `history_load()`, que contém métodos para o carregamento do ficheiro, através do comando `pickle.load(open())`, e o tratamento de dados, em especifico nas colunas `hdata['Date First Defect']`, `hdata['Date Last Defect']` e `hdata['Total TC']` foram modificadas. [Figura 10- Tratamento dos dados da tabela historic_data]

```
def history_load():
    hdata=pickle.load(open(rootfs+'historic_data.sav', 'rb'))

    hdata['Date First Defect'] = hdata['Date First Defect'].dt.date
    hdata['Date Last Defect'] = hdata['Date Last Defect'].dt.date
    hdata['Total TC'] = hdata['Total TC'].astype('int')
    return hdata
```

Figura 10- Tratamento dos dados da tabela historic_data

Após o tratamento dos dados, criou-se uma função que possibilita-se a filtragem dos resultados através de um *click* efetuado pelo utilizador. Foram adicionadas séries de decisões sendo que a variável "c" é responsável pelo *click*, caso uma das condições (exemplo `if c ==" project name"`) fosse verdade, os valores são filtrados de forma ascendente com o seguinte comando `hdata.sort_values(nome da coluna, ascending=True)`. [Figura 11- Tratamento da funcionalidade filtragem da tabela]

```
# initialize var with history data
history_set_model_folder(resource_path("")+"\\")
historic_data = history_load()

# alter sorting order by the click of a column on table history data
def sort_history_rows(c):
    global historic_data
    historic_data=history_sort(historic_data,c)
    table_clear(history_table)
    for index, row in historic_data.iterrows():
        table_insert(history_table,(row["Project Name"], row["Date First Defect"], row["Date Last Defect"],
        row["# Defects"], row["Total TC"], "{0:.1f}%".format(row["Defect Density"]*100)))
```

Figura 11- Tratamento da funcionalidade filtragem da tabela

Antes de se efetuar a inserção dos dados na tabela foi necessário fazer o carregamento dos mesmos, o

carregamento do *dataset* foi realizado através do comando `history_load[]` e estes dados foram guardados na variável `historic_data`,.

Através do comando `table_insert()` foi possível inserir as colunas na tabela. Este ciclo foi inserido dentro da função sort_history_rows(colunas), de forma a possibilitar a filtragem das colunas sem perda de dados. [Figura 12-Inserção dos dados e da funcionalidade de filtragem na tabela]

```
def history_sort(hdata,c):
    if c == "project_name" :
        hdata =  hdata.sort_values(by=['Project Name'], ascending=True)
    elif c == "first_date" :
        hdata = hdata.sort_values(by=['Date First Defect'], ascending=True)
    elif c == "last_date" :
        hdata = hdata.sort_values(by=['Date Last Defect'], ascending=True)
    elif c == "defects" :
        hdata = hdata.sort_values(by=['# Defects'], ascending=False)
    elif c == "defect_density" :
        hdata = hdata.sort_values(by=['Defect Density'], ascending=False)
    elif c == "total_tc" :
        hdata =  hdata.sort_values(by=['Total TC'], ascending=False)
    return hdata
```

Figura 12-Inserção dos dados e da funcionalidade de filtragem na tabela

A criação da estrutura da tabela foi realizada através do comando `create_table`() com todas as variáveis necessárias, e as colunas na tabela *history* foram preenchidas através do comando `history_table.heading`(), aplicando-se sempre a função de filtragem mencionada anteriormente. Figura 13- Criação da estrutura da tabela history_table]

```
# tab: History Data
history_table = create_table(t8, ("project_name","first_date", "last_date","defects","total_tc","defect_density"))
history_table.heading("project_name", text="Project Name", anchor=tk.W, command=lambda c="project_name": sort_history_rows(c))
history_table.heading("first_date", text="Date First Defect", anchor=tk.W, command=lambda c="first_date": sort_history_rows(c))
```

Figura 13- Criação da estrutura da tabela history_table

Por fim é apresentada a tabela "History Data" com o nome do projeto (primeira coluna da tabela), as datas dos primeiros e últimos defeitos (segunda e terceira coluna da tabela), o número de defeitos (quarta coluna da tabela), número de teste cases realizados (quinta coluna) e a densidade dos defeitos (sexta coluna da tabela). [Figura 14- Tabela History Data]

Log Changes Screens Integrations Business Scenarios Test Cases Effort History Data Predictor

Project Name	Date First Defect	Date Last Defect	Defects	Test Cases Defect Density
	2014-08-19	2017-06-28		
	2015-08-05	2021-11-04		
	2016-09-30	2021-11-04		
	2013-08-27	2017-04-20		
	2007-11-16	2012-10-18		
	2018-11-23	2021-11-04		
	2016-05-12	2018-05-14		
	2013-07-04	2021-10-22		
	2016-05-31	2018-05-24		
	2015-07-28	2017-11-29		
	2008-08-07	2015-02-12		
	2008-07-30	2015-07-15		
	2009-11-03	2021-11-02		
	2016-06-27	2020-10-29		
	2015-01-07	2018-11-23		
	2015-01-21	2021-08-04		
	2015-09-01	2016-08-03		
	2018-03-20	2019-11-25		
	2011-09-01	2013-11-07		
	2015-10-02	2017-01-23		

Figura 14- Tabela History Data

Nota: Todas as informações consideradas confidencias, foram ocultadas da tabela em questão.

4.4 Tratamento das imagens

a. Extração de texto em imagens

Uma das tarefas pedidas, era a deteção e extração de texto embebido dentro de imagens nos documentos HLD. Através dos conhecimentos de *Deep Learning* e *Computer Vision*, fez-se uso da biblioteca Pytesseract (explicação no Capítulo 2 - 2.1c), conhecida pela sua capacidade de reconhecimento e leitura de textos embebidos nas imagens para a deteção do texto.

Em adição com o auxílio da biblioteca *OpenCV* (explicação no Capítulo 2 - 2.1c), foi possível fazer o upload das imagens, através do comando `cv2.imread(localização da imagem)`. De seguida o texto foi extraído com o auxilio do comando `pytesseract.image_to_string(imagem escolhida)`. [Figura 15 - Leitura e escrita de texto embebido em imagens]

Com o uso destas bibliotecas foi possível executar esta tarefa de forma simples e rápida. Foi também acrescentada a funcionalidade de tradução, de acordo com a linguagem apresentada em cada documento. Possibilitando a tradução do texto de inglês para português, sendo que todos os textos extraídos eram guardados de acordo com a linguagem pedida.

Figura 15 - Leitura e escrita de texto embebido em imagens

b. Deteção de objetos UI [25] em imagens

O problema proposto nesta tarefa, em complemento com a extração do texto, era a deteção UI de todos os objetos presentes na mesma. Como tal foi necessário o uso de ferramentas de *Deep Learning* que pudessem auxiliar no reconhecimento de objetos.

Previamente, antes da utilização de algoritmos de deteção de objetos, foi realizado um teste de como se poderia realizar a tarefa. A deteção de objetos consiste na identificação e localização de objetos em uma imagem pertencentes a um conjunto de classes predefinidas, como tal foi necessário o uso de algoritmos aplicados à área de *Computer Vision.*

A realização desta tarefa começou simplesmente pela leitura da imagem através da biblioteca OpenCV, e com a opção da biblioteca Pytesseract, *draw rectangle* foi possível efetuar as caixas delimitadoras ou melhor, *bounding boxes*, à volta das imagens e de seguida aplicou-se a designação de cada caixa (*labeling*). [Figura 16-Teste de deteção de objetos GUI]

Este algoritmo não demonstrou ser o mais preciso e eficaz, como se pode verificar na imagem nem todos os objetos GUI foram identificados com sucesso. [Figura 17- Resultado do teste de deteção de objetos GUI]

[25] User Interface é forma de interação entre o utilizador e o sistema, através de telas de exibição e as suas componentes, teclados entre outros. Disponível em What is user interface (UI)? Definition from SearchAppArchitecture (techtarget.com)

```
buttons=0
field =0

for c in cnts:
    x,y,w,h = cv2.boundingRect(c)
    #   if cv2.inRange(hsv, lower_bound, upper_bound):
    if 180 > w > 10 and h > 20 :
        img1= cv2.rectangle(img_box, (x, y), (x + w, y + h), (36,255,12), 2)
        cv2.putText(img1,"Button Detected",(x, y),cv2.QT_FONT_NORMAL,0.5,(36,255,12),1,cv2.LINE_4)
        buttons=buttons+1
    # COLOR RESTRICTION and color == (235,235,235,255)
    elif 250 > w > 50 and h > 20  :
        img2= cv2.rectangle(img_box, (x, y), (x + w, y + h), (250,22,22), 2)
        cv2.putText(img2,"Field Detected",(x, y),cv2.QT_FONT_NORMAL,0.5,(250,22,22),1,cv2.LINE_4)
        field=field+1

print("I have detected "+str(buttons)+ " buttons and "+str(field)+ " fields, in the screenshot.")

#cv2.imshow('canny', canny)
cv2.imshow('image', img_box)
#cv2.imwrite('canny.png', canny)
cv2.imwrite('image.png', img_box)
cv2.waitKey(0)
```

Figura 16-Teste de deteção de objetos GUI

A identificação de cada objeto tem como base, o tamanho (altura e largura), cor e localização na imagem (coordenadas "x" e "y"). Caso haja um objeto com as especificações referidas, este será classificado como "*field*" ou "*botão*".

No caso de objetos do tipo "*field*", estes foram definidos com uma dimensão em que o valor da largura variava entre 50 a 250 pixéis e a sua altura teria de ser superior a 20 pixéis, com uma cor RGB de (250,22,22). Enquanto nos objetos tipo "botão" foram definidos com uma dimensão em que o valor da largura variava entre 10 a 180 pixéis e a sua altura teria de ser superior a 20 pixéis, com uma cor RGB de (36,255,12).

Através destas restrições foi possível o desenho de caixas a volta de cada objeto encontrado, porém o algoritmo não demonstrou ser o mais preciso, uma vez que nem todos os campos foram encontrados.

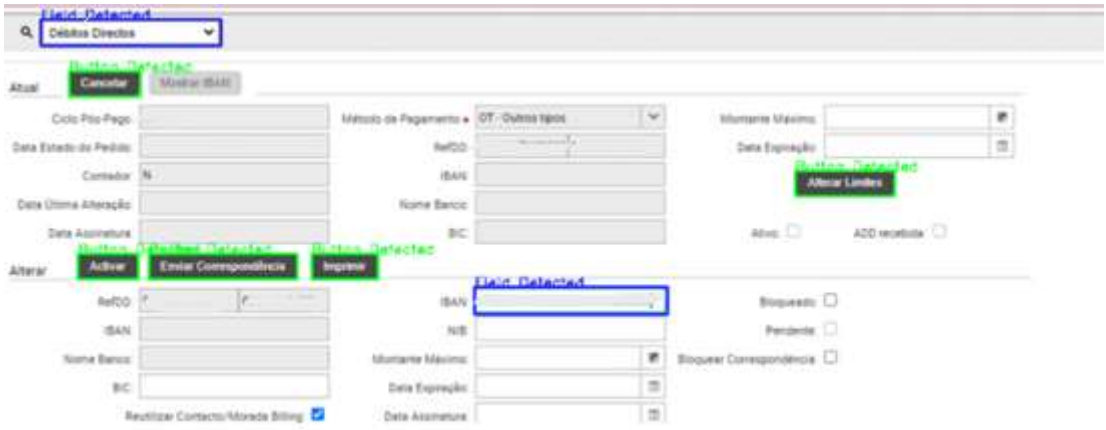


Figura 17- Resultado do teste de deteção de objetos GUI, identificação de objetos do tipo "Field" e "Button"

Como tal foi necessário o uso de algoritmos previamente desenvolvidos, relativos a deteção de objetos que pudessem realizar este método de forma eficaz.

4.5 Algoritmo de reconhecimento/deteção de objetos YOLOv5

Com base nas pesquisas e literaturas obtidas, encontraram-se diversos algoritmos capazes de executar esta tarefa, o algoritmo escolhido para esta tarefa foi o YOLOv5 (versão 5)[26].

Computer Vision e *Deep Learning* são as áreas fundamentais para a deteção e reconhecimento de padrões, um dos algoritmos mais populares é o YOLOv5 (You Only Look Once), desenvolvido pela Ultralytics HUB[27] uma empresa de Machine learning que desenvolve e promove modelos de inteligência artificial. [Figura 18- Exemplo de deteção de pinguins com o algoritmo YOLOv5] [28]

[26] Disponível em YOLOv5 | PyTorch
[27] Disponivel em About - Ultralytics
[28] Imagem obtida em http://www.nsf.gov/

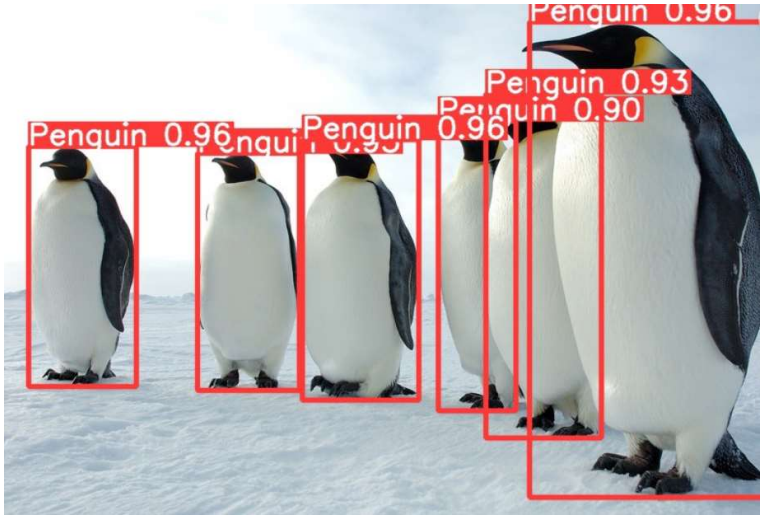


Figura 18- Exemplo de deteção de pinguins com o algoritmo YOLOv5

O YOLO é um modelo de deteção de objetos *open source,* faz uso de redes neuronais de ponta a ponta de modo a detetar objetos em tempo real. O algoritmo processa a imagem inteira, de seguida separa a imagem por partes sendo que todas as partes têm dimensões iguais e por fim prevê as caixas delimitadores e a probabilidade das classes, para cada componente. Os resultados são as caixas delimitadoras que apresentarem maior probabilidade, seguidas pela denominação da sua classe. [Figura 19- Exemplo do funcionamento do algoritmo YOLOv5] [29]

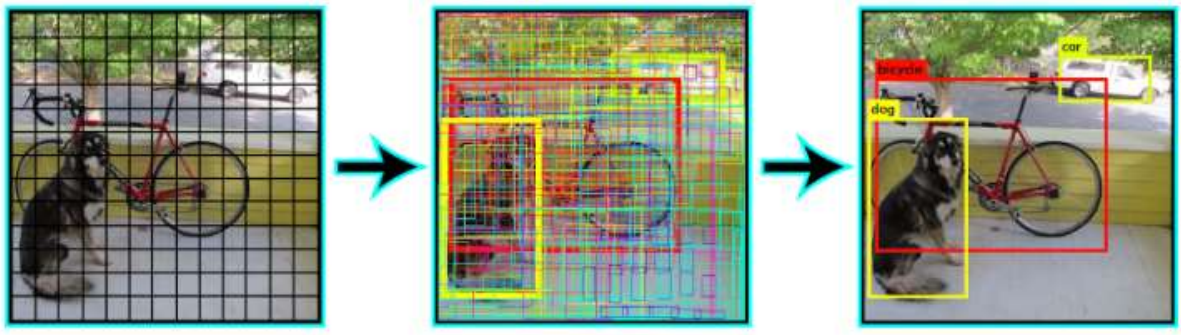


Figura 19- Exemplo do funcionamento do algoritmo YOLOv5

Através deste algoritmo foi possível obter resultados mais precisos comparados com o os resultados obtidos na alínea 4.4 b . Como tal, foi necessário criar um *dataset* com imagens UI. Através do uso da

[29] Imagem obtida em YOLO: Real-Time Object Detection Explained (v7labs.com)

plataforma Roboflow[30] , uma *framework* dedicada ao estudo de *Computer Vision* para a o tratamento, processamento e treino de dados. Com esta *framework* foi possível realizar o anotação de cada imagem de modo que o ficheiro zip, gerado no final, contivesse as imagens e com as respetivas anotações.

Para utilizar a framewrok Roboflow foi necessário seguir os seguintes passos:

1. Upload do *dataset* de imagens ou uso do *dataset* fornecidos pela plataforma;
2. Fornecimento de uma descrição, ou mais bem conhecido por *labeling* de cada imagem;
3. Agrupamento das anotações, de acordo coma descrição especificada;
4. Treino do *dataset*, sendo necessário dividir o *dataset* para treino, validação e teste. Para o estudo em concreto este foi dividido em 66% para testes de treino, 22% para testes de validação e 12% para testes; [Figura 20- Estrutura da divisão de imagens utilizadas para a plataforma Roboflow]
5. Por fim, é fornecido um link (com o ficheiro zip gerado) com o treino e o processamento do *dataset* com todas as imagens agrupadas, com a sua devida descrição pronto para ser usado como o algoritmo YOLOv5.

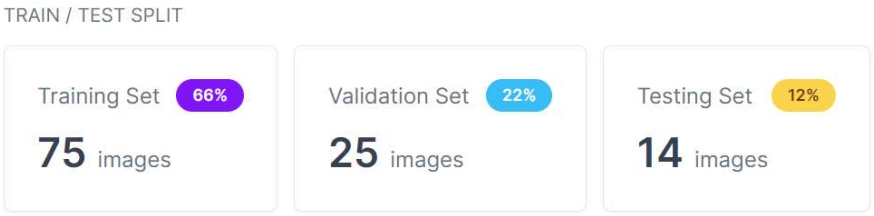


Figura 20- Estrutura da divisão de imagens utilizadas para a plataforma Roboflow

Apos a formatação destas imagens, começou-se por fazer a instalação da *framework* Roboflow no Google Colab e a configuração do algoritmo YOLOv5 [Figura 21 - Uso do *dataset* criado pelo Roboflow]. Por fim, é possível analisar o ficheiro, em concreto, realizar o treino do modelo [Figura

[30] Disponível em Roboflow: Give your software the power to see objects in images and video

22- Treino do modelo].

```
[ ]  !pip install roboflow
     # use thw roboflow dataset created
     from roboflow import Roboflow
     rf = Roboflow(api_key="2B5tmiDKPtTE2dhlFHCa")
     project = rf.workspace("work-mxfrr").project("gui-detector")
     dataset = project.version(6).download("yolov5")
```

Figura 21 - Uso do dataset criado pelo Roboflow

Model Training

```
[ ]  #train the data
     !python train.py --img 416 --batch 16 --epochs 100 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache
```

Figura 22- Treino do modelo

Existem várias métricas utilizadas pelo YOLOv5 para medir a precisão e *performance* de um modelo, subsequentemente são listadas todas as métricas utilizadas[31]:

- Métrica IOU (*Intersection Over Union*) – quantifica o grau de sobreposição entre duas regiões, verifica se as caixas se sobrepõem e caso a haja sobreposição exata essa correspondência será de 1.0 , se não houver será de 0.0. O cálculo desta métrica pode ser efetuado da seguinte forma, a área de intersecção é dividida pela área de união, como demonstra a e quanto maior o IOU mais preciso serão os resultados obtidos. [Figura 23- Fórmula da métrica IOU]

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

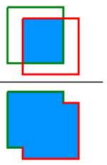


Figura 23- Fórmula da métrica IOU

[31] Fórmulas disponíveis em https://github.com/rafaelpadilla/Object-Detection-Metrics

- Métricas de precisão – mede a precisão das previsões, ou seja, retorna a percentagem de previsões corretas. O cálculo a métrica de precisão é calculada através da seguinte fórmula, em que o acrónimo TP (True Positive) refere-se as deteções corretas efetuadas pelo modelo e o FP (False Positive) as deteções incorretas realizadas pelo detetor. Sendo assim, a precisão é determinada pela divisão entre todas as deteções corretas efetuadas com todas as deteções encontradas.

$$Precision = \frac{TP}{TP + FP\ (False\ Positive)} = \frac{TP\ (True\ Positive)}{todas\ as\ detetções}$$

- Métricas de performance – mede a proporção de positivos reais que foram previstos corretamente. A performance pode ser calculada através da seguinte fórmula, em que o acrónimo FN refere-se as deteções que o algoritmo não conseguiu detetar com sucesso. O *recall* é calculado então pela divisão entre todas as deteções corretas efetuadas com todas as deteções encontradas com sucesso (positivos reais)

$$Recall = \frac{TP}{TP + FN\ (False\ Negative)} = \frac{TP}{positivos\ reais}$$

- Métricas mAP (*mean Average Precision*) – Métrica de avaliação, apresenta a média de AP (área sob a curva do gráfico precisão-recall, este valor é calculado para cada classe separadamente) detetados em todas as classes.

$$mAP = \frac{1}{nº\ de\ classes * sum(AP)}$$

Considera-se um bom modelo aquele que apresenta valores elevados de precisão e performance, este modelo perfeito apresenta valores equivalentes a 1 para as métricas de precisão e *recall*. Com base nas nos resultados obtidos nos gráficos, os valores das métricas encontram-se nos intervalos definidos, contudo o modelo não apresenta ser um modelo perfeito uma vez que os valores da precisão são equivalentes a 0.9 e *recall* 0.25, apresentando assim uma inconsistência no modelo. [Figura 24- Resultado das métricas de precisão obtidas pelo algoritmo]

O aumento do *dataset* pode melhorar os valores da precisão e do *recall*, quanto maior o *dataset* melhor será a probabilidade de perdição. De acordo com a literatura relativa ao algoritmo, de modo a obter um modelo robusto é recomendado o uso de pelo menos 1500 imagens por classe e mais de 10,000 instâncias por classe, reduzindo assim os erros de falsos positivos.

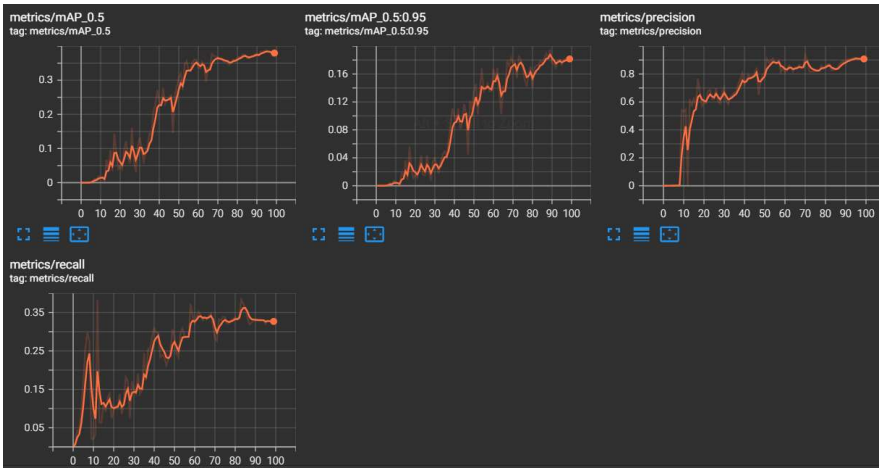


Figura 24- Resultado das métricas de precisão obtidas pelo algoritmo

As imagens utilizadas para este estudo não foram as fornecidas pela empresa, foram utilizadas imagens de um *dataset* fornecido pelo orientador. Na Figura 25 foi detetado com sucesso um objeto do tipo botão com uma precisão de 85%.

A Figura 26 possível demonstra que nem todas as imagens apresentam um bom nível de predição, foram identificados dois objetos com sucesso um com uma precisão de 50% e outra com 69%, contudo a imagem conseguiu detetar dois botões com uma percentagem de 16%. De forma a contornar este problema verificou-se que é possível efetuar uma melhoria aumentando o *dataset* e as suas anotações de modo a melhorar a precisão dos resultados.

Figura 25- Resultado da deteção de objetos feita pelo algoritmo YOlOv5, com uma percentagem de 85% para o projeto CognitiveQA

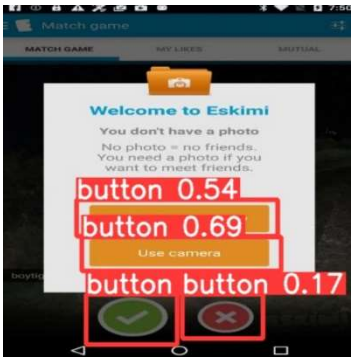


Figura 26-Resultado da deteção de objetos feita pelo algoritmo YOlOv5 para o projeto CognitiveQA

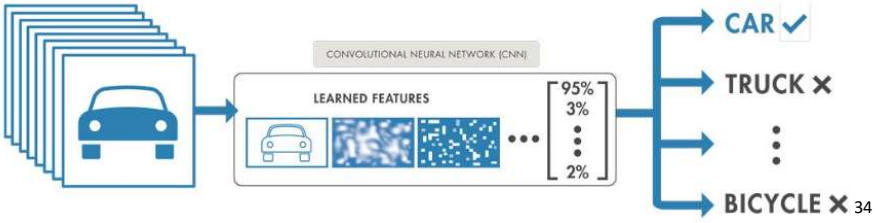
4.6 Algoritmo de reconhecimento/deteção de objetos Detectron2

Outro algoritmo encontrado para o estudo foi o algoritmo Detetctron, versão 2, foi criado pela empresa Facebook no âmbito do estudo de projetos de Inteligência Artificial.[32] Apresenta uma base de código de alta qualidade, alto desempenho para pesquisa de deteção de objetos e flexível, a fim de apoiar a rápida implementação e avaliação de novas pesquisas.

Implementa algoritmos de deteção de objetos e desenvolvido em linguagem Python, a versão 2 inclui algoritmos de deteção de objetos de última geração como CNN, Faster R-CNN, Mask R-CNN entre outros.

[32] Disponível em Detectron2: A PyTorch-based modular object detection library (facebook.com)

Com os avanços realizados nas ares de *Computer Vision* e *Deep Learning* foi possível a criação de algoritmos como CNN Rede Neural convolucional[33], este algoritmo ao receber uma imagem (como input) é capaz de atribuir a importância a vários aspetos da imagem, diferenciando um do outro. [Figura 27- Exemplo do funcionamento do algoritmo CNN]. São particularmente utilizados para encontrar padrões em imagens, deteção de objetos e rostos.



[34]

Figura 27- Exemplo do funcionamento do algoritmo CNN

O Faster R-CNN, faz uso do algoritmo CNN, contudo este algoritmo é mais rápido e apresenta uma rede de proposta de região a volta do objeto deteção, surgindo assim várias propostas para a deteção. O algoritmo é capaz de retornar caixas delimitadoras para cada objeto e seu rótulo de classe com uma pontuação de confiança [Figura 28- Exemplo do uso de Fast R-CNN] [35]

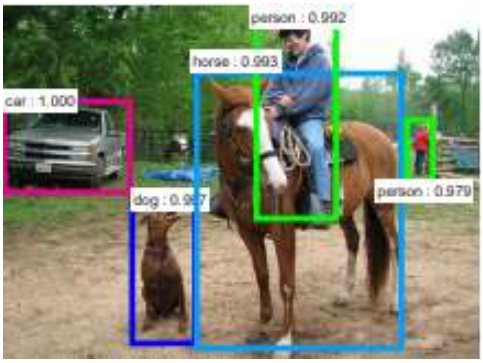


Figura 28- Exemplo do uso de Fast R-CNN

[33] Apresenta uma arquitetura similar às redes neuronais artificias, cada camada da rede é responsável pela extração de informação à entrada dos dados, a informação flui pelas camadas. Estas redes são capazes de simular a capacidade cognitiva humana na identificação de objetos presentes numa imagem e identificação de rostos e indivíduos.

[34] Imagem obtida em What is a Convolutional Neural Network? - MATLAB & Simulink (mathworks.com)

[35] Imagem obtida em 1506.01497.pdf (arxiv.org)

- Mask R-CNN, é um modelo de última geração para segmentação de instâncias, desenvolvido por cima do Faster R-CNN, este algoritmo também deteta objetos, adicionalmente forma uma segmentação para cada instância/objeto encontrado. Classifica cada pixel em conjunto sem diferenciar as suas instâncias. [Figura 29-Exemplo do uso de Mask R-CNN] [36]

Figura 29-Exemplo do uso de Mask R-CNN

Os resultados obtidos pelo algoritmo não foram os esperados, foram encontrados problemas ao longo do uso do *dataset*. ´Deste modo o algoritmo foi descartado para o estudo do projeto, sendo apenas utilizado o algoritmo YOLOv5.

4.7 Comparação entre os algoritmos YOLOv5 e Detectron2

Existem inúmeros algoritmos para a deteção de objetos em imagens e vídeos, um dos mais conhecidos é o Pytesserract que faz uso da biblioteca Opencv, para esta tarefa foram avaliados dois algoritmos.

YOLOv5, "You only look once", baseado em regressões este algoritmo é capaz de prever classes e caixas de texto à volta das imagens. Para além disso, também demonstra a percentagem da previsão realizada. Apresentou ser um algoritmo rápido e com maior exatidão de resultados.

Por outro lado, também foi analisado o algoritmo Detectron 2[37], criada pela empresa Facebook. Esta biblioteca, de próxima geração AI, tem como objetivo ajudar o utilizador a detetar objetos, apresentado

[36] Imagem obtida em 1703.06870.pdf (arxiv.org)
[37] Disponível em GitHub - facebookresearch/detectron2: Detectron2 is a platform for object detection, segmentation and other visual recognition tasks.

bons resultados relativos à predição.

Para a escolha do algoritmo foi preciso ter em conta:

- **O volume de dados e precisão para o treino**

  Caso o *dataset* apresente um volume de dados inferior a 150 o melhor algoritmo a ser utilizado seria o Detectron. Numa fase inicial o volume de dados utilizado não foi superior a 150, contudo como este iria aumentar, foi relevante para o projeto a utilização do algoritmo YOLO. Sendo que a precisão relativamente a este aspeto não diferencia muito.

- **O tamanho do modelo**

  Ambos os algoritmos apresentam desempenhos semelhantes independente que o tamanho do modelo do YOLOv5 seja relativamente menor em comparação ao Detectron.

- **Recursos e tempo de treino**

  O algoritmo YOLOv5 apresenta menos recursos em comparação com o Detectron, devido ao seu menor tamanho de modelo. De modo a tornar os resultados mais precisos e eficientes, foi utilizado o algoritmo YOLOv5, uma veze que utiliza menos recursos.

Concluindo, através dos estudos e testes realizados, verificou-se que a probabilidade de maior precisão seria efetuada através do algoritmo YOLOv5.

Capítulo 5 Conclusão

Ao longo deste capítulo serão descritas as diferentes razões e componentes que levaram ao desenvolvimento deste projeto, os obstáculos encontrados ao longo do desenvolvimento, explicação de trabalhos futuros e possíveis alterações que possam ser realizadas e considerações finais sobre o estágio curricular.

5.1 Trabalho desenvolvido

O trabalho desenvolvido durante o presente estágio, contribuiu positivamente para o avanço do projeto Cognitive QA e para aquisição de conhecimentos.

De forma a integrar no projeto e desenvolver novas capacidades, foi dada a responsabilidade pelo desenvolvimento da ferramenta, nomeadamente na evolução de algumas componentes complexas, responsáveis pelo processo de adição de novos serviços inteligentes à plataforma.

A nível de equipa, os elementos valorizaram o trabalho realizado, através do acompanhamento. a aprovação do desenvolvimento do projeto em termos de código, era submetido e aprovado pela equipa através da plataforma Git, durante as reuniões semanais.

5.2 Principais Obstáculos

A utilização de novas tecnologias e área de *Machine Learning*, fez como que houvesse um atraso no desenvolvimento do projeto, uma vez que era um conceito desconhecido, contudo foi ultrapassado com base no estudo e literaturas encontradas.

Prendendo-se essencialmente também na identificação de objetos nas imagens, com os poucos dados (imagens) apresentados e a utilização de possíveis soluções, impossibilitaram de realizar testes que apresentassem uma melhor precisão nos resultados. No entanto, esta situação foi ultrapassada através do uso de um banco de dados, fornecido pelo orientador, para que os dados possam ser avaliados de uma melhor forma.

5.3 Trabalho Futuro

O trabalho futuro passa melhorar a eficiência dos modelos utilizados para a deteção de objetos. Estes modelos não se encontram no seu nível mais eficiente e a precisão dos mesmos necessita de melhorias, sendo necessário recorrer a mais etapas.

De forma a se conseguir esta evolução, passa por aumentar o tamanho do *dataset,* bem como uma maior diversificação de *label,* com o objetivo de restringir todos os possíveis objetos e para que o modelo consiga identificar com maior precisão possível. Com base nestes dados e algumas melhorias no código de produção é possível evoluir o algoritmo e o treino da máquina, apresentando assim resultados mais coerentes.

5.4 Considerações Finais

De forma a se conseguir alcançar o sucesso final, foi necessária um domínio e utilização de diversos conceitos aprendidos ao longo do curso, como por exemplo, programação Python, utilização de algoritmos de *Machine Learning*, entre outros conceitos.

Para além disso, foi permitido um primeiro contacto com metodologias e técnicas, que permitiu praticar o que, até ao momento, só havia estudado em teoria.

Os objetivos propostos permitiram arrecadar experiência de desenvolvimento a nível empresarial, bem como uma série de conhecimentos e de boas práticas de programação.

Pode assim concluir-se, que os objetivos do estágio foram concluídos e toda a experiência vivida ao longo do presente estágio contribuiu para uma evolução, aumentando assim o seu conhecimento e experiência.

Capítulo 6 Anexos

List of Python Packages

Enchant

Python module, used to check the spelling of a word or give suggestions to correct words, also provides additional features such as antonym and synonym of words, in addition to verifying the existence of a term in a dictionary.
https://abiword.github.io/enchant/

PyEnchant

Python library, used for spellchecking, combines all functionalities of Enchant module.
https://pypi.org/project/pyenchant/

Docx2python

Python library used to extract features from .docx files, such as headers, footers, text footnotes, endnotes, properties, and images to a python object.
https://pypi.org/project/docx2python/

Openpyxl

Python library used to read and write on excel files (xls/xlsm/xltx/xltm).
https://openpyxl.readthedocs.io/en/stable/

Python-docx

Python library used for the creation and update of .docx files.
https://python-docx.readthedocs.io/en/latest/

Docxcompose

Python library used for the concatenation/appending of .docx files.
https://pypi.org/project/docxcompose/

Pandas

Python package used for data analysis provides a fast, flexible and intuitive form to analyse with data.

https://pandas.pydata.org/

Json (JavaScript Object Notation)

a text-based format used for storing and communicating information, can be found as a python package, for managing JSON data.

https://www.w3schools.com/python/python_json.asp

Os

Python module, widely used to interact with the operating system.

https://www.geeksforgeeks.org/os-module-python-examples/

Io

Python module, which allows managing a file's input and output operation. https://docs.python.org/3/library/io.html

Copy

Python library that allows copy operations, these being shallow copies or deep copies. https://docs.python.org/3/library/copy.htmlbase 64

CSV

Common import and export format for databases and spreadsheets, a python library that allows the editing, such as reading and writing on CSV files.

https://docs.python.org/3/library/csv.html?highlight=csv

String

Python module provides string manipulation and a collection of string operations. https://docs.python.org/3/library/string.html?highlight=o%20string#module-string

Zip file

Python module that allows users to read, write, create, append, and list ZIP files. https://docs.python.org/3/library/zipfile.html?highlight=o%20zipfile#module-zipfile

Pprint

Provides an aesthetical and pleasing representation of the data structures, making it easy for users to read. https://docs.python.org/3/library/pprint.html

XML.etree

Python module offers a simple and efficient API for parsing and creating XML data. https://docs.python.org/3/library/xml.etree.elementtree.html?highlight=xml%20etree#modulexml.etree.elementtree

Re

Python library, constituted by a set of regular expression facilities, to simplify the search of a specific pattern on a string.

https://docs.python.org/3/library/re.html

Bibliografia

Referências

Alpaydin, E. (s.d.). *Introduction to Machine Learning.*

Andreou, C. (10 de 04 de 2020). *How to Extract Text from Images with Python*. Obtido de Towards Data Science: https://towardsdatascience.com/how-to-extract-text-from-images-with-python-db9b87fe432b#:~:text=The%20Python%20Library&text=Python-tesseract%20is%20an%20optical,for%20Google%27s%20Tesseract-OCR%20Engine.

Andresen, S. L. (2002). *John McCarthy: Father of AI.* Obtido de ieeexplore: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1039837

Celfocus. (s.d.). *Celfocus*. Obtido de https://celfocus.com/home/

Dick, S. (04 de 10 de 2019). Artificial Intelligence. *Harvard Data Science Review*.

Ertel, W. (2017). *Introduction of Artificial Intelligence.* Springer International Publisher.

Ferreira, J. M. (2020). *INTELIGÊNCIA ARTIFICIAL NA QUALIDADE DE DADOS.* Universidade Nova de Lisboa.

Grosz, B. J. (30 de 11 de 2020). The AI Revolution Needs. *Harvard Data Science Review*, p. 4.

He, K., Gkioxari, G., & Dollar, P. (2018). *Mask R-CNN.* Facebook AI Research (FAIR).

Hugo. (21 de 01 de 2021). *Desktop Elements Detection Using Deep Learning*. Obtido de Desktop Elements Detection Using Deep Learning: https://asiffer.github.io/posts/desktop-elements-detection-using-deep-learning/

Lihi Gur Arie, P. (s.d.). *The practical guide for Object Detection with YOLOv5 algorithm*. Obtido de Towards Data Science.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (31 de August de 1995). *A Proposal For The Darthmouth Summer Research Project on Artificial Inteligence.*

Murugaiyan, S. D. (2012). *WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC.*

Peiyuan Jiang, D. E. (2021). *A Review of Yolo Algorithm Developments.* ratory of E: Key Laboratory of Electronic and Information Engineering (Southwest Minzu University).

Ren, S., He, K., Girshick, R., & Sun, J. (s.d.). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.*

Sarode, K. C. (2020). *ADVANTAGES AND DISADVANTAGES OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING: A LITERATURE REVIEW.* India: IAEME PUBLICATION.

Team, A. F. (22 de 06 de 2021). *How Do You Use Deep Learning to Identify UI Components*. Obtido de Alibaba Cloud: https://www.alibabacloud.com/blog/how-do-you-use-deep-learning-to-identify-ui-components_597859

Willett, R. (01 de 07 de 2019). Engineering Perspectives on AI. *Harvard Data Science Review*.

Wu, Y., Kirillov, A., Massa, M., Lo, W.-Y., & Girshick, R. (2019). *Detectron2: A PyTorch-based modular object detection library.* MetaAI.

Yan, F. (2016). *Deep Learning and its application to CV and NLP.* Edinburgh: University of Surrey.

Zhao, Z.-Q. (2019). *Object Detection With Deep Learning: A Review.*