

## Introdução

Com a cultura *data-driven* as empresas podem aumentar o seu potencial através dos dados, por meio deles a companhia leva análise de informações para outros processos, porém, as empresas precisam lidar com a enorme quantidade de dados, a fim de transformá-los e obter informações e valores. Dessa forma, com a análise de dados é possível avaliar alguns indicadores da empresa, tornando possível a avaliação de diversas esferas do negócio e melhor tomada de decisão, como é o caso desta solução.

Para esta solução, foram usados dados da área de Recursos Humanos, onde teve como objetivo analisar o *turnover* (rotatividade de pessoal), de modo a identificar e compreender o perfil e características do colaborador que deixa a empresa. Para Chiavenato (2014), o turnover é resultado de alguns funcionários saindo e outros entrando no mercado de trabalho. Ainda de acordo com Chiavenato (2014), existem dois tipos de desligamento: o de forma voluntária pelo empregado e o desligamento de iniciativa da organização.

Consequentemente, neste projeto foi coberto todas as etapas de um projeto real de Data Science, sobretudo utilizar os dados para responder às questões abaixo.

- ☐ Quais são os fatores que influenciam para um colaborador deixar a empresa?
- ☐ Como reter pessoas?
- ☐ Como antecipar e saber se um determinado colaborador vai sair da empresa?

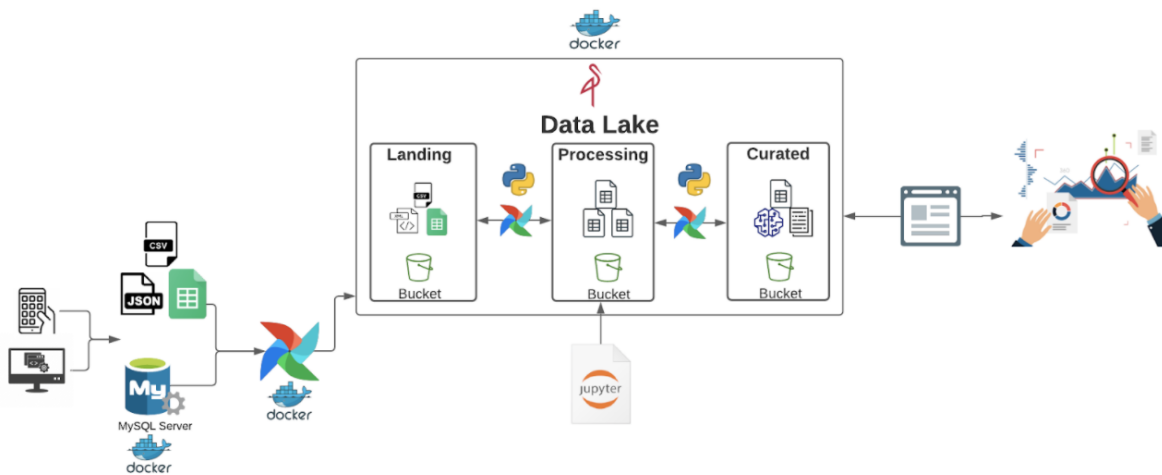
E por fim, disponibilizar recursos para que a empresa consiga realizar a predição para verificar se um colaborador vai ou não deixar a empresa com base em atributos como comportamento e carga de trabalho, nível de satisfação com a empresa e resultados de performance.

Observação: Este projeto foi desenvolvido durante o curso “Human Resource Analytics” da Stack Academy (Stack Tecnologias)

## Arquitetura do projeto

A solução do projeto parte de possíveis sistemas de cliente, podendo ser aplicações desktop ou *mobile*, aplicações essas que são fonte de dados para o desenvolvimento do projeto, a exemplo, arquivos em formatos: json, csv e planilhas de excel, e com esses dados

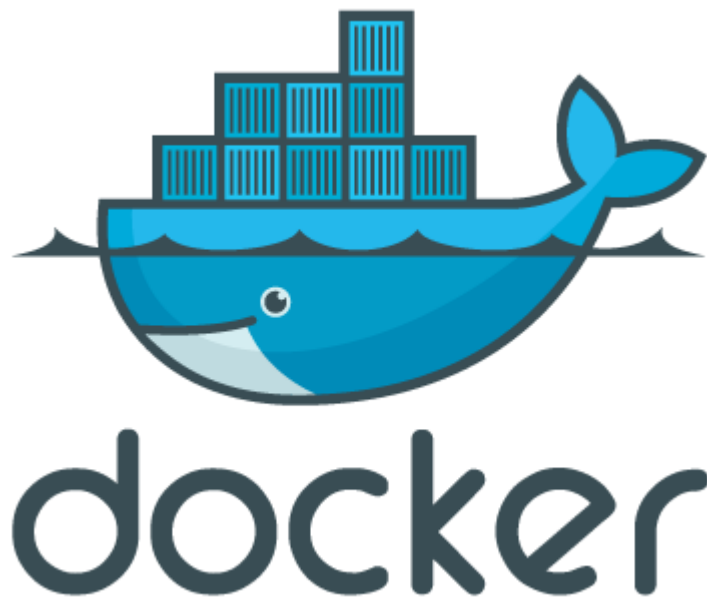
podemos fazer a ligação com o banco de dados.



## Ambiente da solução

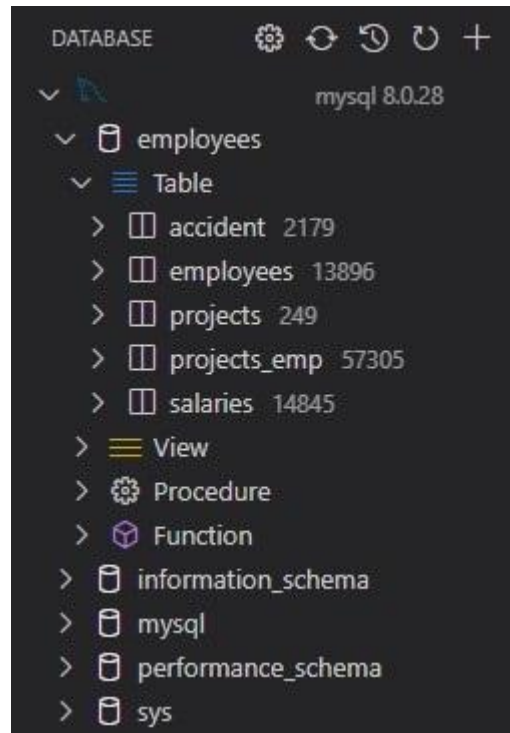
Para execução do projeto foi desenvolvida a estrutura de virtualização de containers para trabalhar com as ferramentas: **MySQL**, **MinIO**, e **Apache Airflow** e, assim, sendo efetuadas todas as dependências necessárias para execução das aplicações,.

solução completa que engloba ciência de dados e engenharia de dados.



## SGBD MySQL

Após a configuração do MySQL, é feita a importação dos dados brutos coletados de planilha excel. A figura abaixo demonstra o conteúdo das tabelas.

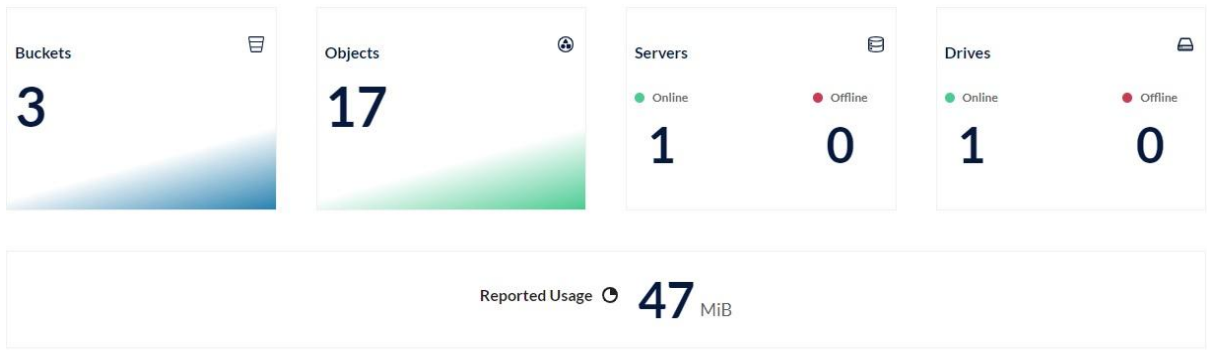


Esses dados são necessários para a de modelagem dos dados e uso do Data Lake com a orquestração/automatização feita com o **Airflow**

**MinIO**



Para o Data Lake da solução usou-se o **MinIO**, servidor de armazenamento de objetos compatível com o protocolo S3, compatível com AWS, escrito em Go.




Dessa forma, conseguimos criar os buckets:

- **Landing** (zona de pouso com dados brutos)
- **Processing** (zona de dados processados)
- **Curated** (zona de dados curados disponível para consumo de aplicações finais)

**curated**  
Created: 2022-02-16T20:37:59Z  
Access: R/W

Manage Browse →



Usage


1.2MiB

Objects

3

**landing**  
Created: 2022-02-16T03:57:00Z  
Access: R/W

Manage Browse →



Usage


46MiB

Objects

7

**processing**  
Created: 2022-02-16T20:37:38Z  
Access: R/W

Manage Browse →



Usage

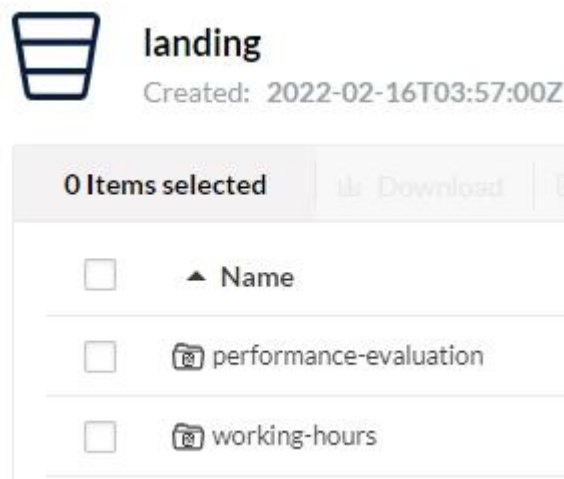
142KiB

Objects

7

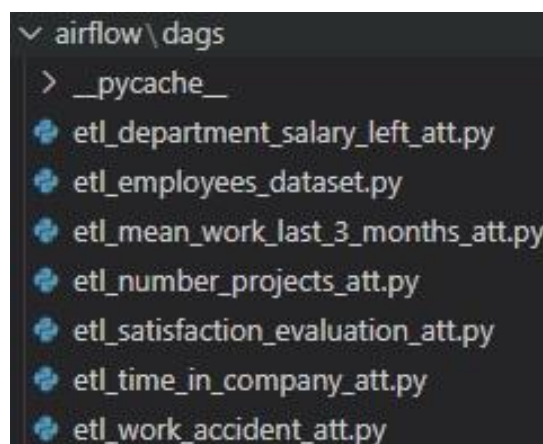
Navegando até o bucket **Lading**, foi feita a carga dos dados e separação dos dados em:

- **working-hours** (registro do ponto eletrônico)
- **performance-evaluation** (avaliação de performance e desempenho)



## Airflow

O Airflow é uma plataforma de gerenciamento de fluxo de trabalho de código aberto para pipelines de engenharia de dados, onde automatizamos as execuções a serem efetuadas, neste projeto utilizamos das DAGS.



Cada DAGS tem como finalidade realizar a carga de arquivos para bucket Processing que está no Data Lake. A tarefa **DAG etl\_employees\_dataset** busca por arquivos em formato parquet (que possui mais eficiência que outros formatos como o csv, consumindo menos memória) na zona processing,

DAGs

All7Active7Paused0

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs1	Schedule	Last Run1	Recent Tasks1	Actions	Links
<div><div></div>etl_department_salary_left_att</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...
<div><div></div>etl_employees_dataset</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...
<div><div></div>etl_mean_work_last_3_months_att</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>4</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...
<div><div></div>etl_number_projects_att</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...
<div><div></div>etl_satisfaction_evaluation_att</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...
<div><div></div>etl_time_in_company_att</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>4</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...
<div><div></div>etl_work_accident_att</div>	Airflow	<div><div>1</div><div></div><div></div></div>	@once	2021-01-13, 00:00:001	<div><div>3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	...

«

<

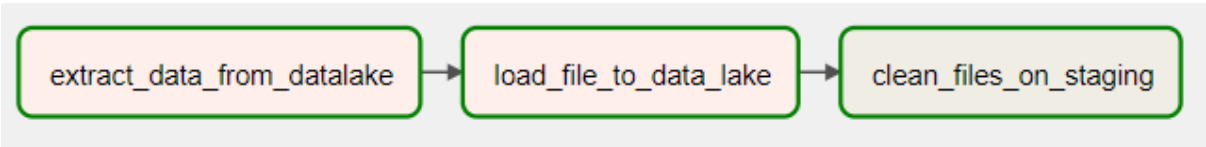
1

>

»

Showing 1-7 of 7 DAGs

Abaixo está a Representação em *graph view* da DAG **etl\_employees\_dataset**



Após a executadas todas as DAGS do **Airflow**, o bucket processing do **MinIO** ficou dessa forma da figura abaixo.



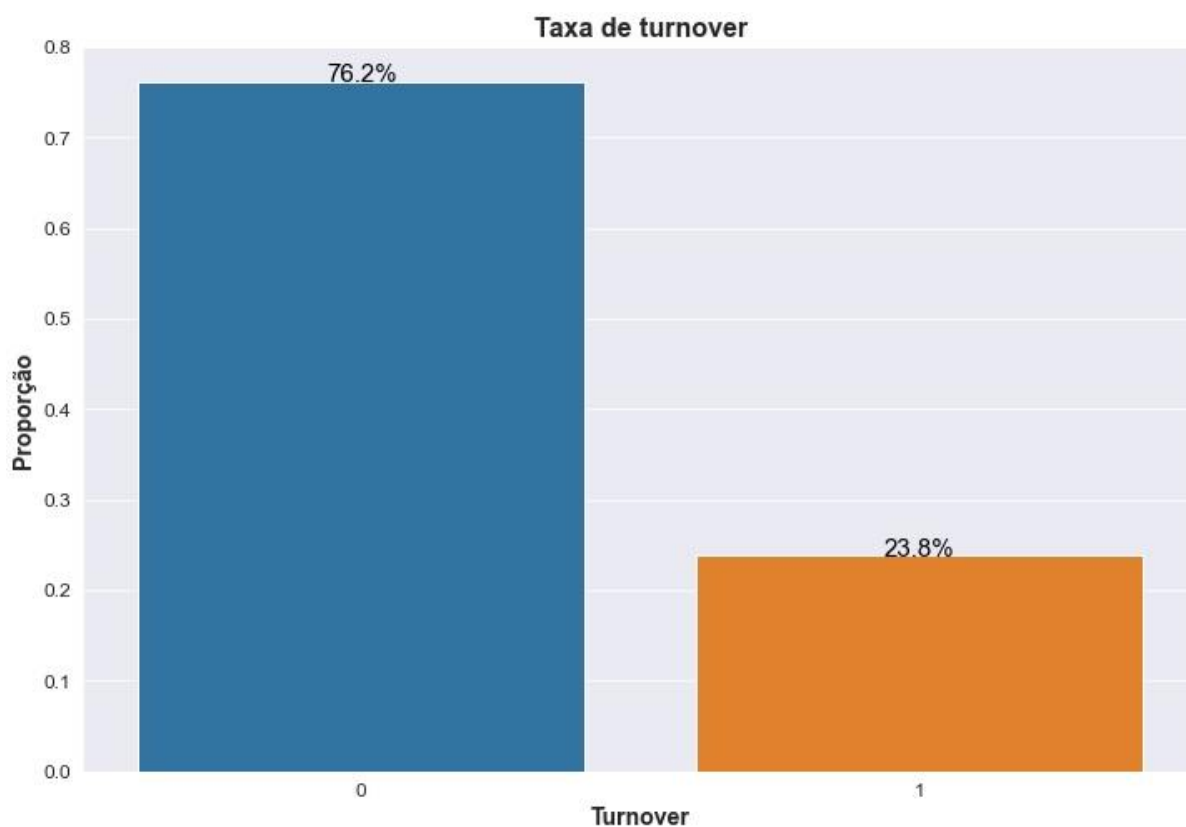
processing

Created: 2022-02-16T20:37:38Z Access: PRIVATE 142 KiB / 7 Objects

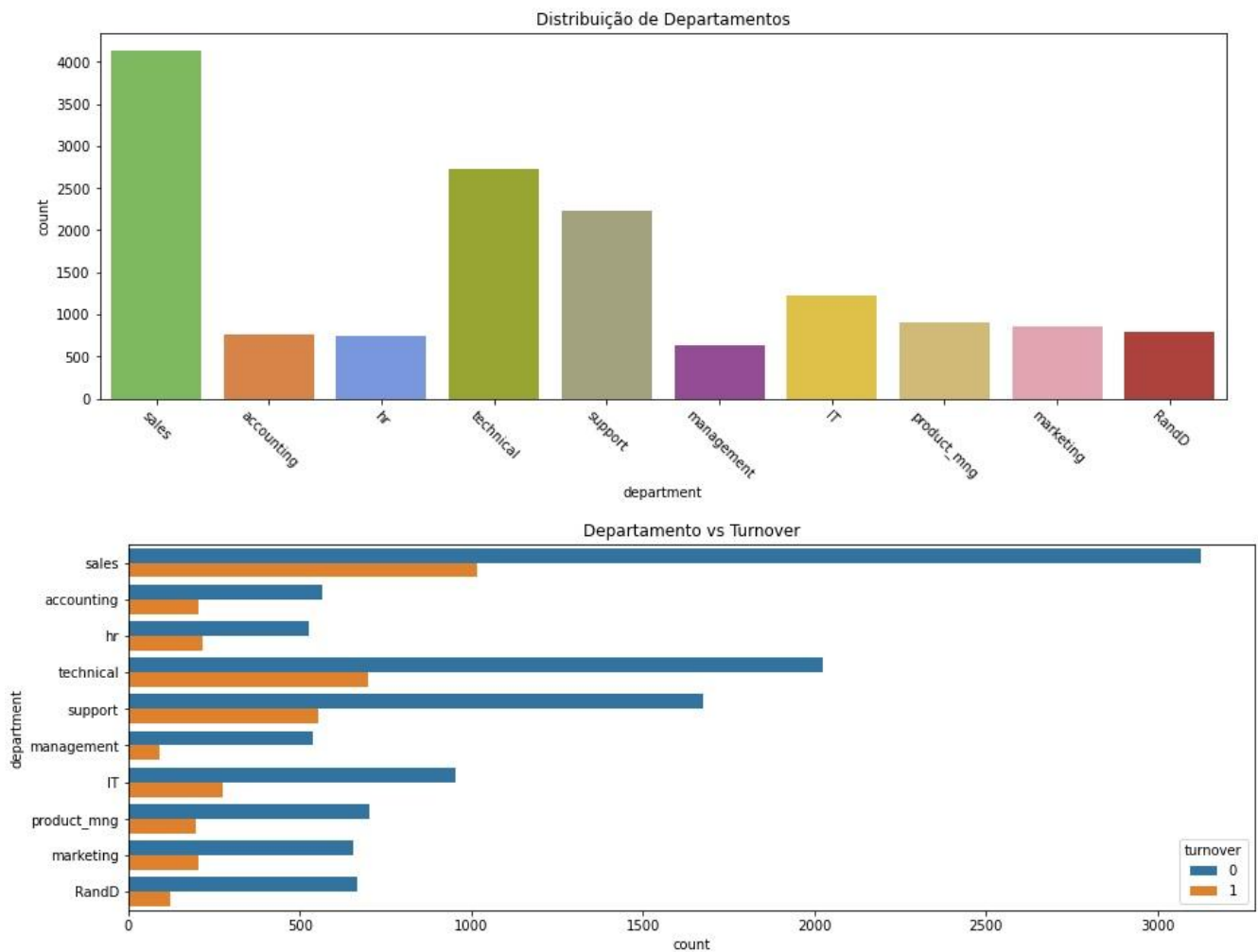
Upload Files

0 Items selected				Download	Share	Preview	Delete	Rewind	New Path	Reload
<input type="checkbox"/>	Name	Last Modified		Size						
<input type="checkbox"/>	department_salary_left.parquet	Wed Feb 16 2022 17:38:45 GMT-0300		10 KiB						
<input type="checkbox"/>	employees_dataset.parquet	Wed Feb 16 2022 19:43:22 GMT-0300		70 KiB						
<input type="checkbox"/>	mean_work_last_3_months.parquet	Wed Feb 16 2022 18:03:47 GMT-0300		17 KiB						
<input type="checkbox"/>	number_projects.parquet	Wed Feb 16 2022 18:51:34 GMT-0300		6.9 KiB						
<input type="checkbox"/>	satisfaction_evaluation.parquet	Wed Feb 16 2022 19:10:54 GMT-0300		28 KiB						
<input type="checkbox"/>	time_in_company.parquet	Wed Feb 16 2022 19:12:04 GMT-0300		6.9 KiB						
<input type="checkbox"/>	work_accident.parquet	Wed Feb 16 2022 19:11:53 GMT-0300		3.4 KiB						

## Análise Exploratória de Dados



De acordo os dados a empresa possui uma taxa de rotatividade de colaboradores de aproximadamente 24% como pode ser visto na figura acima.



Mais detalhes sobre o turnover são apresentados nos gráficos acima, onde podemos chegar à algumas insights:

## Departamento em relação ao turnover

**Resumo:** Vamos ver mais informações sobre os departamentos da empresa.

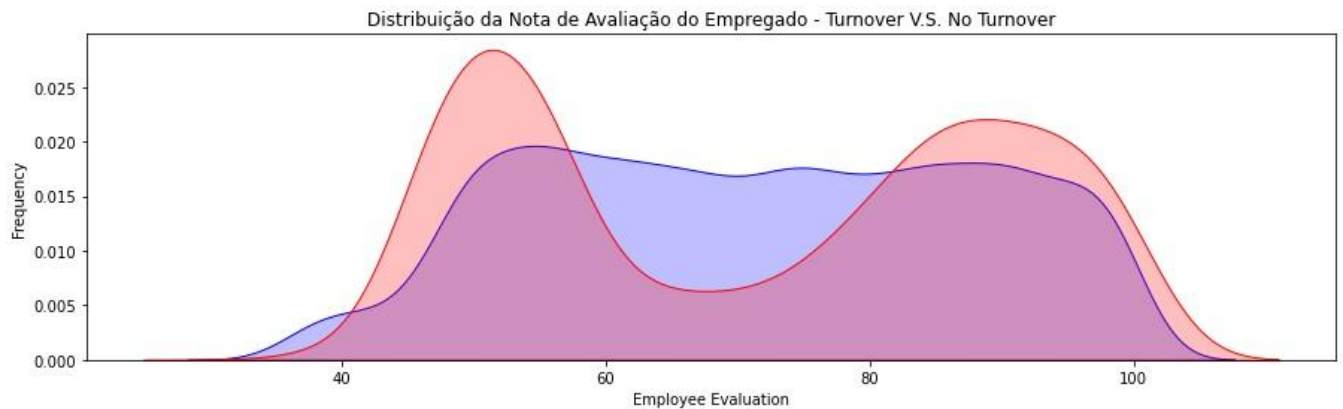
- Os departamentos de **sales** (vendas), **technical** (técnico) e **support** (suporte) são os 3 departamentos com maior índice de turnover.
- O departamento **management** (gestão) tem o menor volume de turnover.

### Questões:

- Será que se examinarmos em profundidade os departamentos que têm maior índice de turnover e o menor pode nos revelar mais informações importantes?
- Qual o salário nestes departamentos?



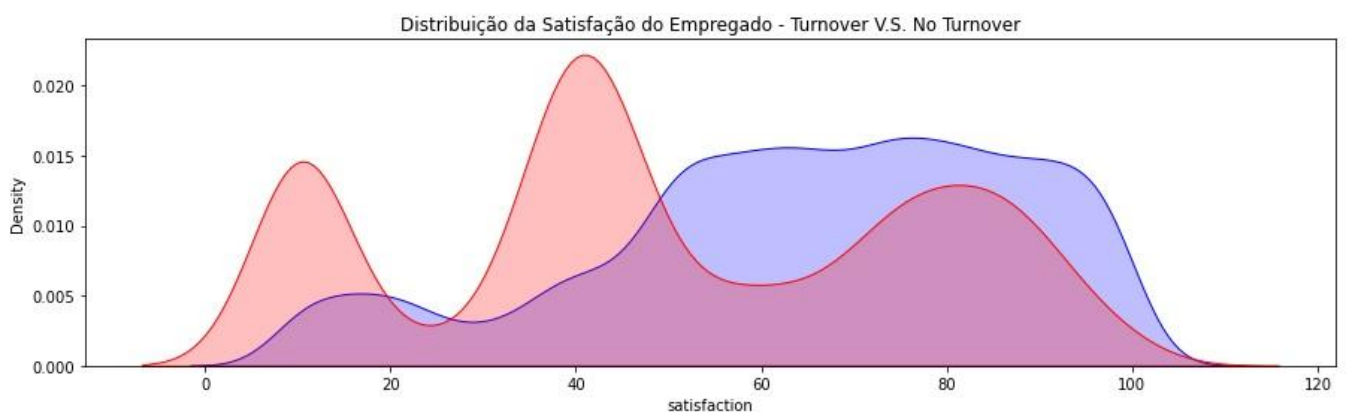
## Nível de Avaliação do Funcionário



De acordo a figura acima, temos um **resumo**:

- Temos uma distribuição bimodal para o conjunto que deixou a empresa.
- Colaboradores com baixa performance tendem a deixar a empresa.
- Colaboradores com alta performance tendem a deixar a empresa.
- O ponto ideal para os funcionários que permaneceram está dentro da avaliação de 60 à 80.

## Nível de Satisfação do Funcionário

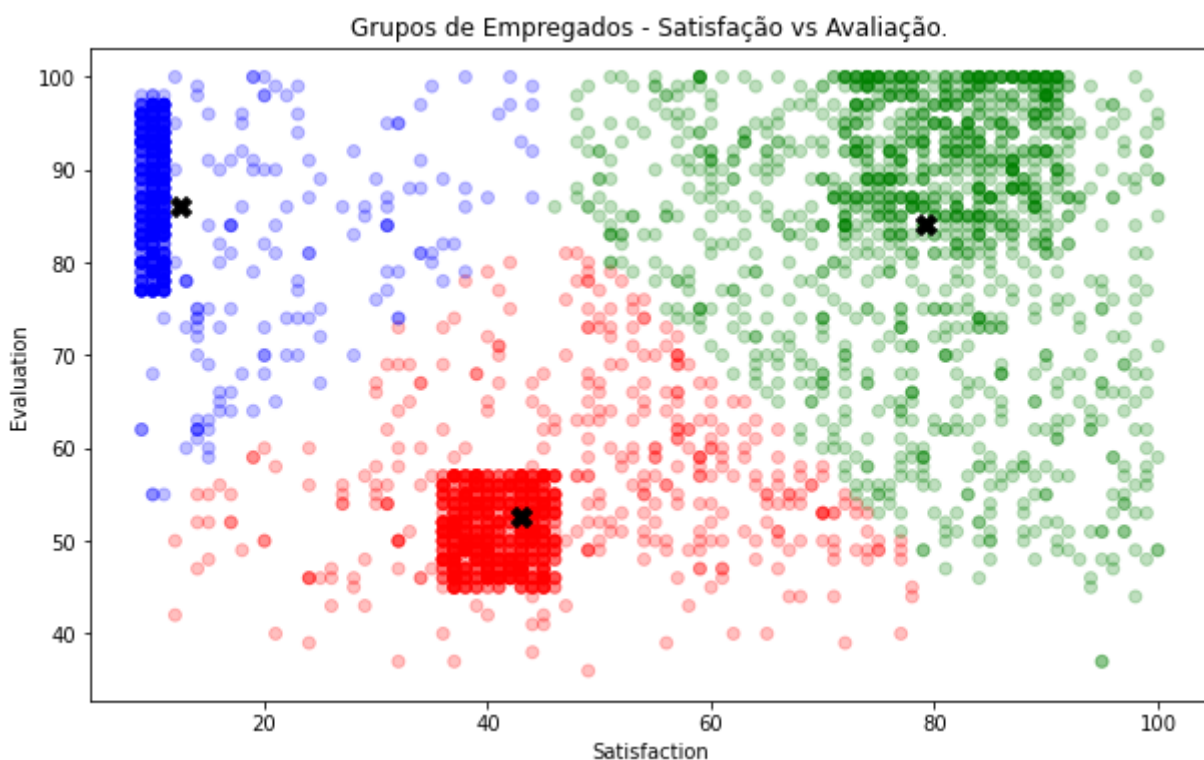


Com o gráfico acima, temos:

- Empregados com o nível de satisfação em 20 ou menos tendem a deixar a empresa.
- Empregados com o nível de satisfação em até 50 têm maior probabilidade de deixar a empresa.

## Clusterização

Com a análise chegamos a 3 *cluster* (grupos) distintos, onde cada grupo é obtido com base nas semelhanças dos dados.



**Cluster 1 (Azul) - (Empregados insatisfeitos e trabalhadores):** A satisfação foi inferior a 20 e as avaliações foram superiores a 75.

- O que pode ser uma boa indicação de que os funcionários que deixaram a empresa eram bons trabalhadores, mas se sentem péssimos no trabalho.

Questões:

- Qual poderia ser o motivo de se sentir tão mal quando você é altamente avaliado?
- Será que está trabalhando muito?
- Esse cluster poderia significar funcionários que estão "sobrecarregados"?

**Cluster 2 (Vermelho) - (Empregados ruins e insatisfeitos):** Satisfação entre 35 à 50 e as suas avaliações abaixo de ~ 58.

Questões:

- Isso pode ser visto como funcionários que foram mal avaliados e se sentiram mal no trabalho.
- Podemos chamar esse grupo de baixo desempenho?

**Cluster 3 (Verde) - (Empregados satisfeitos e trabalhadores):** Satisfação entre 75 à 90 e avaliações superiores a 80.

- O que poderia significar que os funcionários neste grupo eram "ideais".
- Eles amavam seu trabalho e eram altamente avaliados por seu desempenho.

Questões:

- Este grupo pode representar os empregados que deixaram a empresa porque encontraram outra oportunidade de trabalho?
- Poderíamos ter mais do que 3 clusters?

## Machine Learning

### Importância de Features

Converte os atributos categóricos valores numéricos.

In [289]:

```
df["department"] = df["department"].astype('category').cat.codes
df["salary"] = df["salary"].astype('category').cat.codes
```

Separando os conjuntos de dados.

In [291]:

```
target_name = 'turnover'
X = df.drop('turnover', axis=1)
y = df[target_name]
```

Transformando os dados.

In [292]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [293]:

```
scaler = MinMaxScaler()
```

In [294]:

```
X = scaler.fit_transform(X)
```

In [295]:

```
X
```

Out[295]:

Separando os conjuntos.

In [296]:

```
from sklearn.model_selection import train_test_split
```

In [297]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X
    ,y
    ,test_size = 0.2
    ,random_state = 123
    ,stratify = y
)
```

Treinando o algoritmo de árvore de decisão.

In [298]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [299]:

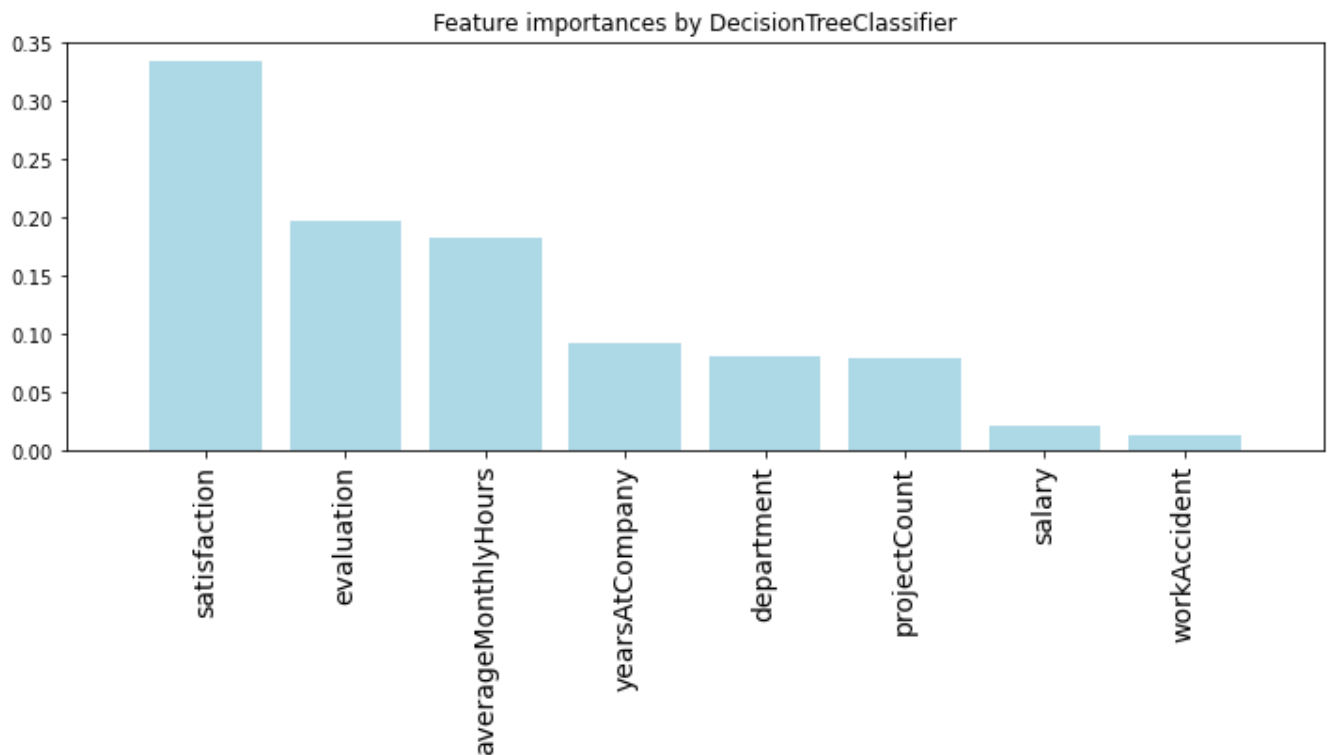
```
dtree = DecisionTreeClassifier()
dtree = dtree.fit(X_train,y_train)
```

In [300]:

```
importances = dtree.feature_importances_
feat_names = df.drop(['turnover'],axis=1).columns
```

In [301]:

```
indices = np.argsort(importances[::-1])
plt.figure(figsize=(12,4))
plt.title("Feature importances by DecisionTreeClassifier")
plt.bar(range(len(indices)), importances[indices], color='lightblue', align="center")
plt.xticks(range(len(indices)), feat_names[indices], rotation='vertical',fontsize=14)
plt.xlim([-1, len(indices)])
plt.show()
```



Podemos ver que no gráfico acima as variáveis que mais contribuem para o modelo são:

**"satisfaction","evaluation","averageMonthlyHours","yearsAtCompany"**

Filtrando apenas os atributos relevantes.

In [302]:

```
X = df[["satisfaction","evaluation","averageMonthlyHours","yearsAtCompany"]]
```

Separando os conjuntos de dados.

In [303]:

```
scaler = MinMaxScaler()
```

In [304]:

```
X = scaler.fit_transform(X)
```

In [305]:

```
X_train, X_test, y_train, y_test = train_test_split(  
    X  
    ,y  
    ,test_size = 0.2  
    ,random_state = 123  
    ,stratify = y  
)
```

In [306]:

```
X_train
```

Out[306]:

Função do modelo de base.

In [307]:

```
def base_rate_model(X) :  
    y = np.zeros(X.shape[0])  
    return y
```

Importando métodos de métrica de avaliação.

In [35]:

```
from sklearn.metrics import roc_auc_score  
from sklearn.metrics import accuracy_score  
from sklearn.metrics import classification_report
```

In [36]:

```
def accuracy_result(y_test,y_predict):  
    acc = accuracy_score(y_test, y_predict)  
    print ("Accuracy = %2.2f" % acc)
```

In [37]:

```
def roc_classification_report_results(model,y_test,y_predict):  
    roc_ = roc_auc_score(y_test, y_predict)  
    classification_report = classification_report(y_test, y_predict)  
  
    print ("\n{} AUC = {}".format(model, roc_))  
    print(classification_report)
```

Análise do modelo de baseline

In [38]:

```
y_predict = base_rate_model(X_test)
```

In [39]:

```
accuracy_result(y_test, y_predict)
```

Accuracy = 0.76

In [40]:

```
roc_classification_report_results("Base Model", y_test, y_predict)  
Base Model AUC = 0.5
```

As acurácias obtidas com os modelos de ML foram:

**Modelo de Regressão Logística.**

Accuracy = 0.77

**Modelo de Árvore de decisão.**

Accuracy = 0.75

**Modelo de Árvore Aleatória (Random Forest)**






Accuracy = 0.83

Abaixo estão a comparação dos modelos utilizando o setup do PyCaret, onde o modelo **gbc** foi o que teve melhor performance.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
<b>gbc</b>	Gradient Boosting Classifier	0.8284	0.8022	0.6883	0.6280	0.6565	0.5426	0.5437	1.0760
<b>lightgbm</b>	Light Gradient Boosting Machine	0.8495	0.7978	0.6571	0.6950	0.6752	0.5774	0.5780	0.8540
<b>ada</b>	Ada Boost Classifier	0.8010	0.7961	0.6995	0.5666	0.6260	0.4925	0.4976	0.4540
<b>rf</b>	Random Forest Classifier	0.8130	0.7815	0.5690	0.6163	0.5916	0.4706	0.4713	1.6280
<b>knn</b>	K Neighbors Classifier	0.7403	0.7597	0.6863	0.4692	0.5572	0.3826	0.3965	0.2900
<b>qda</b>	Quadratic Discriminant Analysis	0.7207	0.7552	0.7487	0.4482	0.5607	0.3744	0.4009	0.0660
<b>et</b>	Extra Trees Classifier	0.8061	0.7525	0.5482	0.6020	0.5737	0.4486	0.4495	1.3700
<b>nb</b>	Naive Bayes	0.7074	0.7105	0.5867	0.4183	0.4883	0.2914	0.2996	0.0440
<b>lr</b>	Logistic Regression	0.6743	0.6913	0.6327	0.3873	0.4804	0.2627	0.2796	2.9120
<b>lda</b>	Linear Discriminant Analysis	0.6772	0.6905	0.6263	0.3893	0.4801	0.2641	0.2798	0.0620
<b>dt</b>	Decision Tree Classifier	0.7342	0.6631	0.5070	0.4487	0.4760	0.2989	0.2999	0.1500
<b>svm</b>	SVM - Linear Kernel	0.6525	0.0000	0.6327	0.3683	0.4648	0.2341	0.2531	0.1400
<b>ridge</b>	Ridge Classifier	0.6772	0.0000	0.6263	0.3893	0.4801	0.2641	0.2798	0.0740

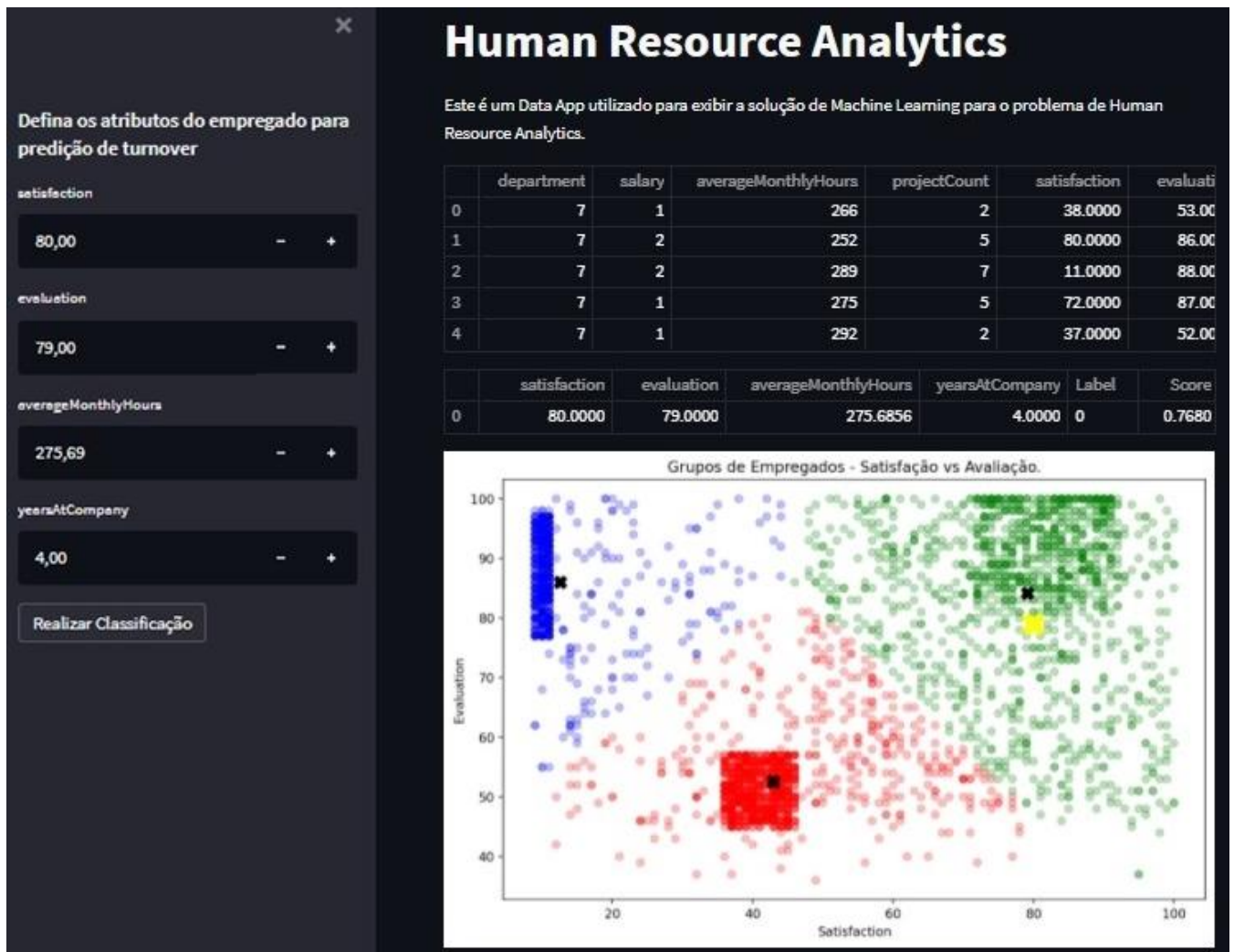
## Dados Curados

Após a finalização do modelo, é feito o armazenamento dos métodos usados no desenvolvimento da solução, sendo salvos na zona de Curated.

	<b>curated</b>	Created: 2022-02-16T20:37:59Z	Access: PRIVATE	1.1 MiB - 3 Objects	<a href="#">Rewind</a>	<a href="#">Reload</a>	<a href="#">Upload</a>
<	 curated						
<input type="checkbox"/>	Name	Last Modified		Size			
<input type="checkbox"/>	 cluster.joblib	Thu Feb 24 2022 20:19:30 GMT-0300		15 KiB			
<input type="checkbox"/>	 dataset.csv	Thu Feb 24 2022 20:19:30 GMT-0300		381 KiB			
<input type="checkbox"/>	 model.pkl	Thu Feb 24 2022 20:19:30 GMT-0300		771 KiB			

## Disponibilização do App da Solução

Para disponibilização da solução foi usado o framework **Streamlit**, tecnologia para criação de Data App com script em Python. Abaixo pode ser visto o deploy da solução, onde se tem as variáveis para realização da predição.



## Conclusão

Através desse projeto foi possível praticar e implementar conceitos importantes da Ciência e Engenharia de Dados e propor uma solução para um problema latente e recorrente de qualquer empresa que é a retenção de talentos através da Análise de Dados de Recursos Humanos. Como um processo de melhoria contínua podemos desenvolver uma automação para executar não só o pipeline de coleta e transformação de dados como automatizar os passos da etapa de Machine Learning e Deploy.

Caso tenha interesse em consultar o projeto, ficam aqui meu o repositório do projeto e meu LinkedIn para possível contato:



- LinkedIn: <https://www.linkedin.com/in/luiscarlos-almeida/>
- Repositório Github:  
[https://github.com/luiscals1/Data\\_Science/tree/main/human-resources-analytics](https://github.com/luiscals1/Data_Science/tree/main/human-resources-analytics)

## Referências

CHIAVENATO, Idalberto. **Gestão de Pessoas**: o novo papel dos recursos humanos nas organizações. 4. ed. Barueri: Manole, 2014.