

Tutorial para o Desenvolvimento do Projeto

Infraestrutura do Projeto: Tecnologias utilizadas



1. Preparação do ambiente

1.1 Faça download do Anaconda no site:

Realizar o download do arquivo .sh

<https://www.anaconda.com/products/individual#Downloads>

No terminal Linux realizar a instalação do Anaconda a partir do arquivo .sh

<https://docs.anaconda.com/anaconda/install/linux/>

- For Python 3.7, enter the following:

```
# Include the bash command regardless of whether or not you are using the Bash shell
bash ~/Downloads/Anaconda3-2020.05-Linux-x86_64.sh
# Replace ~/Downloads with your actual path
# Replace the .sh file name with the name of the file you downloaded
```

Após instalar o Anaconda, criar o atalho na área de trabalho.

<https://www.joelotz.com/blog/2021/creating-a-launch-shortcut-for-anaconda-on-ubuntu.html>

<https://linux.how2shout.com/create-anaconda-navigator-desktop-shortcut-ubuntu-20-04-18-04/>

1.2 Faça download do Visual Studio Code (baixar o pacote .debian)

Construção do APP para deploy

<https://code.visualstudio.com/download>

1.3 Instalação e Configuração do SQLite e SQLiteStudio/DBBrowser for SQLite

BD para fazer consultas SQL na base de dados

<https://www.sqlite.org/download.html>

1.3.1 Primeiro instale o SQLite via terminal

<https://cloudinfrastructureservices.co.uk/how-to-install-db-browser-for-sqlite-in-ubuntu-server-20-04/>

1.3.2 Instalação do SQLiteStudio

<https://sqlitestudio.pl/>

Baixar o arquivo com extensão .run :

<https://github.com/pawelsalawa/sqlitestudio/releases/download/3.4.4/SQLiteStudio-3.4.4-linux-x64-installer.run>

Ir nas propriedades do arquivo e marcar o arquivo como um programa
pesquisar : install file .run ubuntu

Duplo click no arquivo para iniciar a instalação.

1.3.3. Instalação do DB Browser for SQLite

O DB Browser for SQLite pode ser instalado através do repositório do Linux ou através do link abaixo.

<https://github.com/pawelsalawa/sqlitestudio/releases/download/3.4.4/sqlitestudio-3.4.4.tar.xz>

1.4 Instalação e Configuração do Apache Airflow

<https://airflow.apache.org/>

Utilizado para fazer os processos de orquestração, automação, ETL

Realizar a instalação do Apache Airflow no prompt do Anaconda.

pip install apache-airflow

Navegue até o diretório airflow e crie o diretório dags.

Criar o arquivo: data_pipeline.py e depois copiá-lo para dentro do diretório dags

1.5 - Git e GitHub

publicar o App no repositório

Criar uma conta no github: <https://github.com/>

1.6 Streamlit

Tecnologia utilizada para criar data Apps

<https://streamlit.io/>

Realizar a instalação do streamlit no prompt do Anaconda

Ir no diretório do anaconda, dentro da pasta bin do anaconda instalar o streamlit

```
./pip3 install streamlit
```

```
pip install streamlit
```

Realizar o cadastro (solicitar convite para o cadastro na plataforma streamlit) para permitir publicar o data app na web.

<https://share.streamlit.io/>

Logar com a conta do github para conectar o github com o streamlit.

O streamlit vai buscar seus fontes no github

2. Entendimento do Negócio e Explorando a base de dados no SQLiteStudio

Entendimento do Negócio: Aluguel de Imóvel

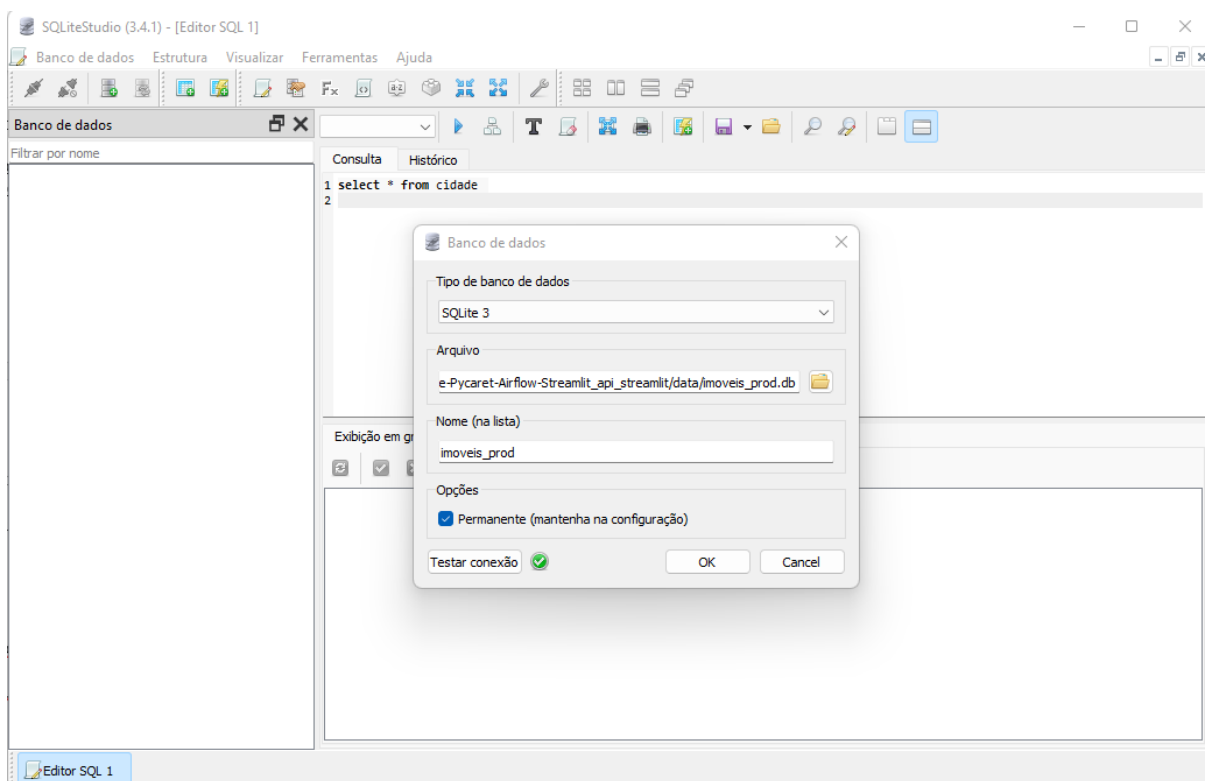
Qual é o preço ideal de aluguel para um imóvel ?

Objetivo: Desenvolver um modelo de ML para realizar a predição do preço ideal do aluguel de um imóvel.

Abra o SQLiteStudio.

Clique em **Database** em seguida clique em **Add Database**

Em seguida adicione os bancos de dados **imoveis_prod** e **imoveis_dw** clicando no link da pasta abaixo e localizando o arquivo, veja o exemplo abaixo:



Comandos SQL:

-- EXPLORANDO A BASE DE DADOS

SELECT * FROM CIDADE;

SELECT * FROM ESTADO;

SELECT * FROM IMOVEIS;

SELECT * FROM PROPRIETARIO;

-- CONTANDO A QUANTIDADE DE IMOVEIS.

SELECT COUNT(*) FROM IMOVEIS;

-- FILTRANDO REGISTROS

SELECT AREA

,NUM_QUARTOS

,NUM_BANHEIROS

,VALOR_ALUGUEL

,VALOR_IPTU

FROM IMOVEIS

WHERE NUM_QUARTOS > 2;

--CONTANDO A QUANTIDADE DE IMOVEIS POR CIDADE.

SELECT CODIGO_CIDADE,COUNT(*) AS "QTD" FROM IMOVEIS

GROUP BY CODIGO_CIDADE;

-- UNINDO TABELAS

SELECT CIDADE.nome as 'cidade'

,ESTADO.nome as 'estado'

,IMOVEIS.AREA as 'area'

,IMOVEIS.num_quartos

,IMOVEIS.num_banheiros

,IMOVEIS.num_andares

FROM IMOVEIS INNER JOIN CIDADE

ON IMOVEIS.CODIGO_CIDADE = CIDADE.CODIGO

INNER JOIN ESTADO

ON CIDADE.CODIGO_ESTADO = ESTADO.CODIGO;

```
-- SELECIONANDO O DATASET FINAL
SELECT CIDADE.NOME as 'cidade'
,ESTADO.NOME as 'estado'
,IMOVEIS.AREA as 'area'
,IMOVEIS.NUM_QUARTOS
,IMOVEIS.NUM_BANHEIROS
,IMOVEIS.NUM_ANDARES
,IMOVEIS.ACEITA_ANIMAIS
,IMOVEIS.MOBILIA
,IMOVEIS.VALOR_ALUGUEL
,IMOVEIS.VALOR_CONDOMINIO
,IMOVEIS.VALOR_IPTU
,IMOVEIS.VALOR_SEGURO_INCENDIO
FROM IMOVEIS INNER JOIN CIDADE
ON IMOVEIS.CODIGO_CIDADE = CIDADE.CODIGO
INNER JOIN ESTADO
ON CIDADE.CODIGO_ESTADO = ESTADO.CODIGO;
```

3. Construindo o Modelo de Machine Learning

3.1 Conexão do python no BD imoveis_prod.db

Abra o Anaconda Navigator.

Abrindo o Jupyter Notebook: Clique em launch para abrir o jupyter notebook

Ao abrir navegue até o diretório criado para o projeto, por exemplo: notebooks

Dentro do diretório notebooks, crie o seguinte arquivo deverá conter o notebook:

aula01.ipynb

Executar o Código para realizar a Conexão com o banco de Dados SQLite

3.2 Criar o Data Pipeline (DAG - Directed Acyclic Graph) para consumir os dados em produção

Extract -> com o arquivo: imoveis_prod.db converte para imoveis.csv

Transform -> No arquivo .csv converte dados categóricos, renomeia cidades, realiza o tratamento de dados.

Load -> Exporta o arquivo .csv com os dados transformados para o arquivo de banco de dados imoveis_dw.db

Subindo o Airflow e criando as Dags:

Crie o arquivo: `data_pipeline.py` e salve-o na pasta dags do Apache Airflow e rode os comandos do Airflow:

Abra o terminal do anaconda:

-- Iniciando o banco de dados

airflow db init

-- cria o usuário (**Substitua o colchete com o seu nome**)

airflow users create --username admin --password admin --firstname [nome] --lastname [sobrenome] --role Admin --email [seu e-mail]

-- Iniciando o servidor web

airflow webserver

-- Iniciando o agendador de tarefas

airflow scheduler

a. Mova os arquivos `data_pipeline.py` para o diretório:




































































































































































`/home/gustavo/Documentos/api_streamlit/dags`

b. Aguarde uns 5 minutos e acesse o console do airflow no endereço:

<http://localhost:8080/>

c. Faça o login com usuário admin e a senha admin.

d. Ao clicar em Dags deve aparecer as dags criadas como na imagem abaixo:

<div>  DAGs Security Browse Admin Docs </div> <div>18:22 UTC </div>									
<div> <div>DAGs</div> <div> <div> <div>All 7</div> <div>Active 0</div> <div>Paused 7</div> </div> <div>Filter DAGs by tag</div> <div>Search DAGs</div> </div> </div>									
DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links		
 etl_department_salary_left_att	Airflow	  	@once	  	            	  	...		
 etl_employees_dataset	Airflow	  	@once	  	            	  	...		
 etl_mean_work_last_3_months_att	Airflow	  	@once	  	            	  	...		
 etl_number_projects_att	Airflow	  	@once	  	            	  	...		
 etl_satisfaction_evaluation_att	Airflow	  	@once	  	            	  	...		
 etl_time_in_company_att	Airflow	  	@once	  	            	  	...		
 etl_work_accident_att	Airflow	  	@once	  	            	  	...		

3.3 Realizar análise exploratória dos dados (Aula 02)

3.4 Remoção de outliers e construção do Modelo de Machine Learning (Aula 03)

Nesta sessão iremos construir o modelo de ML, usando PyCaret se desejar, persistir o modelo no formato .pkl

3.5 Pycaret (Aula 04)

Salvar o modelo no disco (dump): model-final.pkl

3.6 Construção do app -> Utilizar o VSCode para criar o arquivo app.py

Abrir o VSCode e criar o arquivo app.py (este arquivo que irá fazer o deploy da aplicação).

Subir este código para o github para publicar a aplicação no git e rodar a partir do github.

no prompt do anaconda, ir no diretório da api_streamlit :

C:\Users\gusta\Meu Drive\Colab Notebooks\DML\api_streamlit (Windows)
/home/gustavo/Documentos/api_streamlit (Linux)

rodar o comando: streamlit run app.py

Se ocorrer erro no streamlit ao tentar fazer o deploy do app, remover a biblioteca protobuf (4.19) e reinstalar a versao (4.21)

pip uninstall protobuf

pip install protobuf==4.21

3.7 Deploy do Modelo de Machine Learning localmente

Iremos utilizar o framework Streamlit para construir o data app para consumir o modelo de machine learning desenvolvido.

No VSCode abrir o arquivo app.py dentro da pasta app

Para executar o app.py localmente, abrir o terminal linux, ir na pasta app e executar o comando:

```
streamlit run app.py
```

Será exibido uma interface web com o app rodando localmente

3.8 Deploy do Modelo de Machine Learning na nuvem

Preparar o arquivo com as bibliotecas python requeridas para rodar a aplicação na web

No VSCode editar o arquivo: requirements.txt

3.9 Subindo o código do streamlit para o git_hub

<https://share.streamlit.io/>

Conectar usando a conta do github

Criar o repositório no github com os arquivos necessários para o app:

app.py #página web a ser carregada pelo streamlit

requirements.txt #libs que devem ser instaladas no ambiente em nuvem do streamlit

data/imoveis.csv

data/imoveis_dw.db

data/imoveis_prod.db

model/modelo-final.pkl

3.10 Carregar o app e depois gerar uma URL para o app na nuvem:

<https://gustavowillam-api-streamlit-app-kso71z.streamlit.app/>