**VISÃO GERAL DO PROJETO - ANALISE DE CRÉDITO 2024**

**RESUMO**

EMPRESA: MONTADORA DE VEICULOS

TODAS AS ETAPAS DO PROJETO

**1 - Necessidades do cliente**

**2 - Dados**

**3 - EDA - Analise Exploratória de Dados**

**4 - Pré-processamento**

**5 – Criação do modelo de Machine Learning**

**6 - XAI - Inteligência Artificial Explicável**

**7 – Criação de API (Interface de Prog. de Aplicativos) + Criar UI (Interface do Usuário)**

**8 - Deploy**

**9 - Possíveis melhorias**

**1 – Necessidades do cliente**

- Sistema de concessão de crédito

- Avaliar a concessão de crédito

- Performance mínima: Precisão mínima 70%, Esperado 80%

- Recall mínimo 70%, Esperado 75%

- Limiar de decisão: 50%, ajustável

- Tecnica de Explicabilidade (XAI)

- Servida através de API (Interface de Programação de Aplicativos)

- Tela UI (Interface Usuário)

App web (Streamlit) + Flask, sendo a infraestrutura: AWS - EC2.

**2 – Dados**

**2.1 - Tabelas: Clientes, Pedido de crédito, Parcelas Crédito, Produtos financiados**

**2.2 - Informações das tabelas:**

- Clientes: ClienteID, Nome, CPF, Email, Profissao, TempoProfissao, Renda, TipoResidencia, Escolaridade, DataNascimento, Dependentes, EstadoCivil, Telefone, Score

- PedidoCredito: SolicitacaoID, ClienteID, ProdutoID, ValorSolicitado, ValorTOtalItem, NumeroParcelas, DataSolicitacao, Status

- ParcelasCredito: SolicitacaoID, NumeroParcelas, ValorParcela, DataVencimento, DataPagamento, Status

- ProdutosFincanciados: ProdutoID, NomeComercial, PrecoTabela

**2.3 - Relacionamentos:**

Clientes/ClienteID -> PedidoCredito/ClienteID

PedidoCredito/SolicitacaoID -> ParcelasCredito/SolicitacaoID

PedidoCredito/ProdutoID -> ProdutosFinanciados/ProdutoID

**2.4 - Identificar atributos com valores semânticos - Atributos importantes**

- Tabela Clientes: Profissao, TempoProfissao, Renda, TipoResidencia, Escolaridade, DataNacimento, Dependentes, EstadoCivil, Score

- Tabela PedidoCredito: ValorSolicitado, ValorTotalBem

- Tabela ParcelasCredito: Status

- ProdutosFinaciados: NomeComercial (do produto)

**2.5 - Cardinalidade - Categorias**

Profissão: Cientista de Dados, Médico, Dentista, Contador...

TipoResidencia: Outros, Alugada, Própria,

Escolaridade: Ens. Fundamental, Ens. Médio, Pósou Mais, Superior

EstadoCivil: Divorciado, Casado, Solteiro, Viuvo

Score: Baixo, MuitoBom, Bom, Justo

Produtos: AgileXplorer, e outros

**2.6 - Variáveis Numéricas**

- Não há problema de cardinalidade

- Porém devemos considerar: Valores nulos e Outliers (valores discrepantes)

**2.7 -Criada a Tabela: Modelo**

Tabela Modelo: Profissao, TempoProfissao, Renda, TipoResidencia, Escolaridade, Score, Idade, Dependentes, EstadoCivil, Produto, ValorSolicitado, ValorTotalBem

**2.8 - Acesso a base de dados**

- Conexão remota via PGADMIN

- Credenciais do banco de dados

dbname: 'novadrivebank'

user: 'etlreandonlybank'

password: 'novadrive376A@'

host: '159.223.187..110'

- Acessando tabelas do banco de dados

- Utilizei Query Tool do PGADMIN

- Criei consulta SQL, para desnomalização dos dados

- Dados foram normalizados após a execução do SQL

**3 - EDA – Analise Exploratória de Dados**

**3.1 – Criação do Ambiente:**

- Python3.7 ou superior

- VSCode

- Virtual Env (ambiente virtual)

- Instalação das bibliotecas necessárias – requirements.txt

fuzzywuzzy==0.18.0: Calcula a similaridade entre duas strings usando diferentes algoritmos, como Levenshtein distance e Jaro-Winkler distance.

pandas==2.2.1: Fornece estruturas de dados e ferramentas de análise para manipulação de dados eficiente em Python.

Scikit-learn==1.4.1.post1: Uma biblioteca de aprendizado de máquina com algoritmos para classificação, regressão, clustering e outras tarefas.

joblib==1.3.2: Uma biblioteca que permite a persistência de objetos Python e a execução de tarefas em paralelo.

PyYAML==6.0.1: Uma biblioteca para carregar e salvar dados em formato YAML.

psycopg2-binary==2.9.9: Um adaptador para o banco de dados PostgreSQL.

numpy==1.26.4: Uma biblioteca para computação numérica em Python, com suporte para arrays e matrizes multidimensionais.

tensorflow==2.16.1: Uma biblioteca de aprendizado profundo para construir e treinar modelos de redes neurais.

shap==0.45.0: Uma biblioteca para explicar modelos de aprendizado de máquina, ajudando a entender como eles chegam a suas previsões.

flask==3.0.3: Um framework web para construir aplicativos web em Python.

python-Levenshtein==0.25.1: Uma biblioteca para calcular a distância de Levenshtein entre duas strings.

streamlit==1.32.1: Uma biblioteca para criar aplicativos web interativos para ciência de dados e aprendizado de máquina.

matplotlib==3.8.3: Uma biblioteca para criar gráficos e visualizações de dados em Python.

seaborn==0.13.2: Uma biblioteca de visualização de dados baseada no Matplotlib, com uma interface mais alta e gráficos mais bonitos.

----------

- Utilizei e configurei o arquivo: config.yaml, para mascarar usuário e senha ao acesso do banco de dados, mas adicionei o nome dele no arquivo .gitignore, para questão de segurança.

Mascarar usuário e senha para acesso ao banco de dados no projeto. Não envie esse arquivo para o servidor, inclua no arquivo .gitignore

- Arquivo contendo dados de logins: PostgreSQL, denominado config.yaml

- Arquivo contendo a query para consulta dos dados, denominado: const.py

- Arquivo, com as informações necessárias para conexão ao banco de dados: utils.py

**3.2 - EDA**

**3.2.1 - Busca obter informações sobre:**

3.2.1.1 - Variações

3.2.1.2 - Anomalias

3.2.1.3 - Distribuição dos dados

3.2.1.4 - Domínio

**3.2.2 - Buscando:**

3.2.2.1 - Distribuição / Outliers (dados discrepantes)

3.2.2.2 - Nas (dados faltantes) / Nulos

3.2.2.3 – Dados fora do domínio

**3.2.3 - Atributos Categóricos**

3.2.3.1 - Gráficos de Barras. Mostrando a frequência por categoria. Indicando valores fora do domínio.

3.2.3.2 - Valores nulos

**3.2.4 - Atributos Numéricos**

3.2.4.1 - Distribuição: Boxplot, Histograma, Resumo Estatístico

3.2.4.2 - Valores nulos

Gráfico Histograma - Gráfico de colunas

Gráfico Boxplot - Possui informações de: Outlier, Máximo, Q3, Mediana, Q1, mínimo

Q1 (Primeiro Quartil): Corresponde ao valor abaixo do qual se encontram 25% dos dados. É o limite inferior da caixa no boxplot.

Q3 (Terceiro Quartil): Corresponde ao valor abaixo do qual se encontram 75% dos dados. É o limite superior da caixa no boxplot.

**3.2.5 - Conclusões EDA**

3.2.5.1 - Dados inválidos: Profissão, Tempo de Profissão e Idade

3.2.5.2 - Dados Nulos: Profissão, Tipo de Residência

**4 - Pré-processamento**

- Converter Tipos

- Tratamento de Nulos

- Erros de digitação

- Tratamento de Outliers – Tratamento de dados discrepantes

- Feature Engineering - Engenharia de Recursos

- Divisão em treino e Teste

- Normalização

- Codificação

- Seleção de atributos

**4.1 - Tratamento de nulos**

- Dados categóricos: Moda

- Dados numéricos: Mediana

4.2 - Erros de Entrada - Dados Categóricos

- Vamos usar o domínio da categoria: valores possíveis

- Não é possível fazer função genérica e universal

**4.3 - Outliers**

- Abordagem estatística: Desvio padrão ou Intervalo Interquartil

- Intervalo Interquartil: (IIQ) é uma medida estatística que indica a dispersão dos dados em um conjunto.

- Abordagem de negócio: Domínio do atributo

--Tempo de profissão: 0-70

--Idade: 0-110

**4.4 - Engenharia de Atributos**

- Codificação de Variáveis Categóricas

- Normalização e Padronização

- Tratamento de Valores Ausentes

- Interactions

**4.5 - Divisão de Dados em Treino e Teste**

**4.6 - Normalização de DADOS**

**4.7 - Padronização (Z-Score)**

**4.8 - Normalização (Min-Max)**

**4.9 - Codificação Categórica ou Categorical Encoding**

**4.9.1 - Label encoding – Codificação de rótulo**

**4.9.2 - Seleção de Atributos**

**5 - Modelo**

**5.1 - Funções de ativação: Relu e Sigmoid**

**5.2 - Função de Perda**

**5.3 - Otimizador**

**5.4 – Dropout - Desistentes**

**5.5 - Matriz de Confusão (Classificação)**

**5.6 - Acurácia**

**5.7 - Precisão**

**5.8 - Recall**

**5.9 - Técnicas para melhorar o modelo**

**5.9 - Versão com mudanças**

**6 - Explicabilidade**

**6.1 - White Box / Black Box**

**- Black Box**

-- Modelos não podem ser interpretados

--- Redes Neurais Artificiais

--- Modelos baseados em grupos

--- Maquina de Vetor de Suporte

**- White Box**

-- Modelo de fácil interpretação/entendimento

--- Regressão Linear

--- Arvores de Decisão

--- Modelos baseados em Regras

**6.9.7 - Visão mais detalhada**

- Escolaridade <= 0.00: -0.195433

- 0.00 < tiporesidencia <=1.00: 0.0737412

- produto > 5.00: 0.044722

**7 - API**

- Base de Dados

- Scalers

- Normalizadores

- Feature Selector

- Modelo

- Relatórios

**7.2 - O que vamos fazer?**

- Criar aplicação Flask

- Criar um script python para testar o Flask

- Criar uma aplicação WEB Streamlit que vai consumir a API

**7.3 - Flask**

**7.4 - Streamlit**

**8 - Deploy**

- Etapas

-- Criar Repositório no Github

-- Configurar gitignore

-- Fazer push para repositório Remoto

-- Configurar instancia do EC2

-- Fazer clone do Repositório

-- Rodar API e aplicação WEB

-- Testar com aplicação Local

-- Testar com aplicação Remota

**8.6 - Criando VM AWS**

- Faça login e acesse a pagina inicial do AWS

- Pesquise sobre: EC2

- Na aba da esquerda, procure e clique no campo: Instances

- Selecione o campo: Launch as Instance

- Para o campo: Name and tags, informe: Novadrivebankvm

- Para o campo Application and OS Images (Amazon Machine Image), selecione a imagem: Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

- Architecture: 64-bit

- Instance type: t2.medium

- Crie um novo par de chave para login

- Informe um nome para a sua par de chave

- Selecione em Key pair type: RSA

- Private key file formato: .pem

- Clique em: Create key pair

- Salve a chave em um local, para uso posterior.

- Selecione no campo: Key pair (login), a chave gerada na tela anterior.

- No campo: Network settings, deixe selecionado: Create security group para Firewall (security groups)

- Deixe marcado: Allow SSH traffic from, Allow HTTPS traffic from the internet e Allow HTTP traffic from the internet

- Clique em: Launch instance, para que a instancia seja criada.

- Volte para aba: Instances e atualize a pagina

- Aguarde 2 min para que a instancia fique com o status: rodando...

- Clique em: Instance ID da instancia desejada

- Uma janela será aberta com as informações da instancia, desça a página e procure por: Security

- Clique no campo: Security groups

- Uma janela será aberta, no campo: Inbounds rules, clique em: Edit inbounds rules

- Clique em: Add rule

- Para o campo: Type, selecione: Custom TCP.

- Para o campo: Port range, informe 5000, que é a porta do Flask

- Para o campo: Source, selecione: Anywhere-IPv4

- Crie outra regra

- Para o campo: Type, selecione: Custom TCP.

- Para o campo: Port range, informe 8501-8505, que é a porta do Streamlit

- Para o campo: Source, selecione: Anywhere-IPv4

- Clique em: Save rules

- Volte para a tela principal da sua instancia

- Clique no botão: Connect

- Clique na aba: SSH client

- Copie o comando do campo: Exemple, para acessar o diretório remotamente de sua instancia na AWS por SSH

- Abra o prompt de comando do Windows, em sua máquina local

- Acesse o diretório onde foi copiada a chave para acesso a sua instancia no AWS.

- Copie o comando para acesso da sua instancia AWS, cole no prompt, dentro do diretório onde encontra-se sua chave

- Confirme a operação com: Yes

- Com acesso a console de sua instancia. digite: sudo apt update

- Abra o Github, no repositorio do seu projeto, clique no campo: Code, copie a url do seu projeto (HTTPS)

- Volte para a máquina virtual e digite: git clone URL\_COPIADA\_DO\_GITHUB\_REPOSITORIO

- Aperte a tecla ENTER

- Digite: ls para listar e verificar se o diretório do repositorio Github, encontra-se na instancia AWS

- Digite: cd novadrivebank/, para acessar o diretório

- Digite: python3 --version, para verificar a versão do python

- Crie o arquivo config.yaml

- Digite: touch config.yaml

- Digite: ls, para listar o diretório

- Digite: nano config.yaml, para editar o arquivo

- Instale o pip na console de sua instancia, basta digitar: sudo apt install python3-pip

- Selecione Y e dê ENTER

- Será aberta uma janela de configuração, clique na tecla TAB, não será alterado nada

- Continuará selecionado: polkit.service, cliquem em OK com a tecla ENTER

- Digite: ls para listar o diretório

- Instale os: requirements.txt

- Digite: pip3 install -r requirements.txt

- Crie os objetos do modelo, digite: python3 modelcreation.py

- Para que a console do terminal da instancia não fique travada/bloqueada, pois temos que rodar duas aplicações, digite o seguinte comando: nohup python3 api.py &

- A API já está rodando por padrão na porta: 5000

- Comando para verificar se a aplicação está rodando: sudo lsof -i : 5000

- Digite para rodar a aplicação streamlit: nohup python3 -m streamlit run webapp.py &

- Volte para a página AWS, instancia da aplicação, procure pelo campo: Public IPv4 DNS e copie o endereço

- Cole o endereço em uma nova aba do seu navegador e adicione: :8501 (dois pontos e 8501)

- Pronto, aplicação rodando totalmente online.

**8.7 - Testando API (rodando na AWS)**

- Acesse: config.yaml no VSCode

**8.8 - Parando e terminando instancia no EC2/AWS - !!!Cuidado!!!**

- Não deixe rodando instancia no AWS

-- Gera custos

- Para uma instancia

-- Ela continua no servidor, mas não gera custos

- Terminar uma instancia

-- Ela não constará mais no servidor, será excluída, não gera custos

**9 - Considerações finais e possíveis melhorias**

**9.1 - Tratar labels não treinados**

- Mesmo tratando o input(não selecionado na amostra)

- Como mitigar

-- Técnica de amostragem que garanta a presença dos labels no ntreino

-- Data Augmentation

-- Continual Learning

-- Novel Detection

**9.2 - Técnicas de CI/CD**

- Git Action

- Jenkins

- Ferramentas Especializadas no ciclo de vida do Modelo:

-- Mlflow

-- Sagemaker (AWS)

-- Azure Machine Learning

**9.3 - Segurança / Autenticação**

- Aplicação WEB

- API/FLASK

**9.4 - Testes Unitários**

- Pytest