Documentação dos modelos CRISP-DM

1. Entendimento do Negócio

1.1 Objetivo do Projeto

Este projeto tem como objetivo o desenvolvimento de uma ferramenta com um alto ganho analítico e de inferência para outras áreas, baseados numa arquitetura robusta e de alta interoperabilidade. É esperada a criação de um pipeline de Big Data, utilizando a infraestrutura em cloud, para o gerenciamento de dados. O foco central reside na necessidade da criação de análises estatísticas detalhadas sobre áreas de foco do cliente, incluindo geografia e canais de atendimento.
Consequentemente, o resultado final será um sistema que permite uma avaliação precisa do potencial de consumo em categorias específicas, possibilitando que os vendedores estabeleçam metas estratégicas e implementem ações táticas direcionadas ao desenvolvimento de categorias ou canais de distribuição. Além disso, o projeto visa criar um infográfico adaptável que simplifique a análise, facilitando assim o processo de tomada de decisões informadas e estratégicas.

2. Entendimento dos Dados

2.1 Origem dos Dados

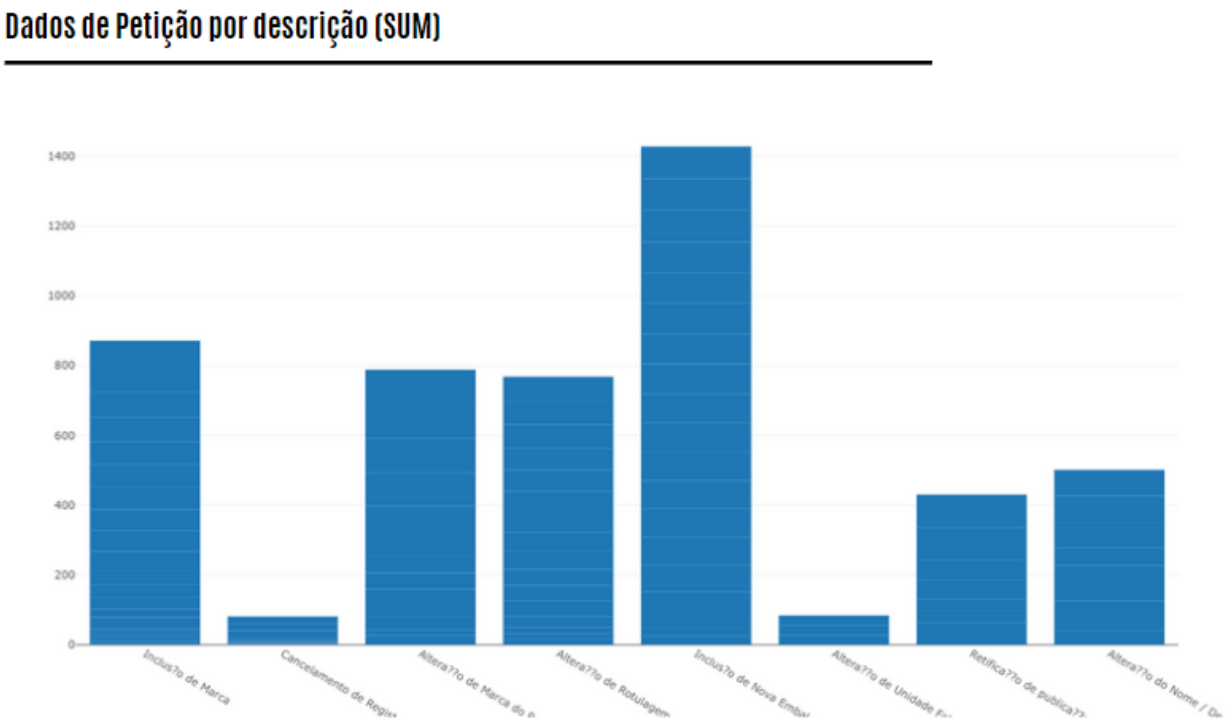
Os dados para o projeto foram coletados de diversas fontes, incluindo o IBGE, BACEN, POF, ANVISA, CNPJ e uma API fornecida por um parceiro que disponibiliza informações sobre as vendas de determinados produtos, juntamente com os CNPJs correspondentes. A ampla variedade de fontes visa proporcionar uma análise abrangente e fundamentada, abordando aspectos demográficos, econômicos, de consumo, regulatórios e de desempenho comercial. Essa abordagem multifacetada visa oferecer uma compreensão mais completa do cenário em questão, permitindo tomadas de decisões estratégicas mais informadas.

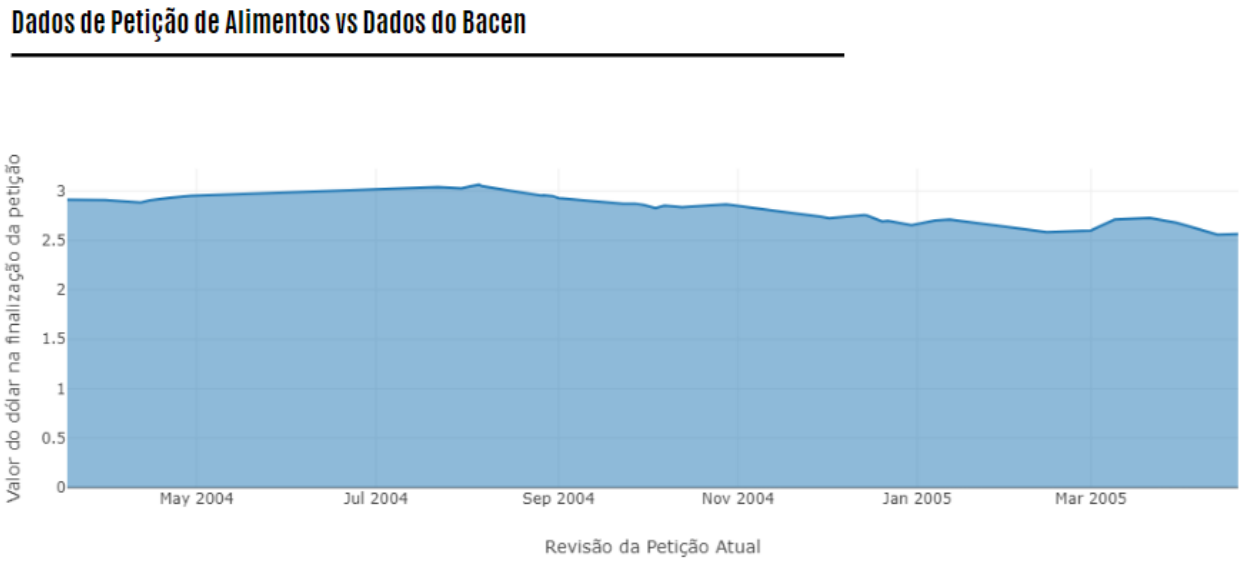
2.2 Exploração Inicial dos Dados

Na primeira exploração dos dados foi possível identificar alguns dados que poderiam ser relacionados entre as diferentes fontes, permitindo uma análise mais detalhada e completa do conjunto de dados selecionados.

Houve também a criação de gráficos iniciais para a visualização do que estava sendo feito e demonstrar insights que podem ser obtidos.

Exemplos:





3. Preparação dos Dados

Como estamos tratando de um volume massivo de dados, é necessário uma série de preparações para que os resultados obtidos sejam da melhor qualidade possível para ter o potencial de agregar ao cliente. Para isso nós seguimos as seguintes técnicas de preparação de dados:

3.1 Seleção dos dados

Visando selecionar os conjuntos de dados que mais agregam com a necessidade do parceiro, foi necessária a escolha dentre os diversos conjuntos de dados disponíveis. Pensando nos possíveis insights obtidos, os conjuntos de dados selecionados pelo grupo foram os dados do BACEN, POF, CNPJ, dados da Anvisa e os dados da API do cliente.

3.2 Pré-processamento

Para viabilizar as visualizações e a construção do modelo Ensemble, foi necessário realizar um pré-processamento em todas as tabelas utilizadas. Esse procedimento visa evitar resultados enviesados e incompatíveis, que não poderiam ser devidamente considerados. O pré-processamento envolve a transformação dos dados necessários e a limpeza de todas as informações que precisavam ser adaptadas para que sejam ingeridas da melhor maneira pela visualização e modelo.

Exemplos de código do pré-processamento:

```
  def process_table(self, remove_duplicates=False, null_value_replacement=None, column_mapping=None,
column_value_mapping=None):
      if remove_duplicates:
          self.df.drop_duplicates(inplace=True)

      if null_value_replacement:
          self.df.fillna(null_value_replacement, inplace=True)

      if column_mapping:
          self.df.rename(columns=column_mapping, inplace=True)

      if column_value_mapping:
          for column, mapping in column_value_mapping.items():
              self.df[column] = self.df[column].map(mapping)

      return self.df
```

```python
def convert_dates(df, column):
          for column in df.columns:
              if df[column].dtype == ['object', 'datetime64[ns]']:
                  try:
                      df[column] = pd.to_datetime(df[column])
                  except ValueError:
                      pass
          return df
```

```python
def encode_columns(df):
  df = df.applymap(lambda x: x.encode('utf-8') if isinstance(x, str) else x)
  df = df.replace(r'\W', '', regex=True)
  return df
```

Ao realizar o pré-processamento, são aplicadas transformações e limpezas necessárias para adaptar os dados à melhor forma de ingestão tanto pela visualização quanto pelo modelo Ensemble. Essa adaptação é vital para garantir que as informações sejam interpretadas corretamente e que o modelo construído seja robusto e preciso. Quando os dados passam por esse processo, tornam-se mais aptos para serem carregados no Redshift (data warehouse). O Redshift é otimizado para lidar com grandes volumes de dados e consultas complexas, mas para aproveitar ao máximo sua eficácia, é imperativo que os dados estejam organizados, limpos e prontos para serem consumidos pela plataforma. Assim, o pré-processamento não apenas contribui para a qualidade das análises e visualizações, mas também assegura que o Redshift seja alimentado com dados coesos e bem-estruturados, proporcionando uma base sólida para análises subsequentes e otimizando o desempenho geral do ambiente de data warehouse.

3.3 Seleção das Features

A seleção de features, ou características, é um componente fundamental no desenvolvimento de modelos ensemble. A importância dessa seleção reside em diversos aspectos para o sucesso do ensemble.

Em primeiro lugar, a redução da dimensionalidade é um benefício significativo da seleção de features. Em conjuntos de dados de alta dimensionalidade, a eliminação de características irrelevantes, redundantes ou colineares acelera o treinamento do modelo e previne o overfitting (mesmo que neste sentido nosso modelo ainda esteja um pouco “overfitado”), melhorando assim a generalização. Além disso, a inclusão de features relevantes pode otimizar a performance do ensemble. Da mesma forma que features irrelevantes podem introduzir ruído nos modelos individuais, comprometendo a precisão das previsões. Por isso, a seleção cuidadosa das features melhora a qualidade das informações fornecidas aos modelos e, consequentemente, a eficácia do ensemble como um todo.

Como a prevenção do overfitting é importante, decidimos como premissa, ter features bem pensadas, tendo em vista que a presença de muitas características pode levar a modelos individuais que se ajustam demais aos dados de treinamento, prejudicando a capacidade de generalização. A seleção de features relevantes ajuda a evitar esse problema, permitindo que o ensemble se concentre nas informações mais relevantes.

Além disso, a robustez e estabilidade do ensemble são fortalecidas pela escolha cuidadosa das features. A remoção de características irrelevantes ou redundantes reduz a sensibilidade a variações nos dados de entrada, tornando o modelo mais resistente e capaz de generalizar de maneira confiável para novos dados.

Com bases nestes aspectos e visando o melhor resultado possível e utilizando das análises previamente mencionadas, as seguintes features foram selecionadas:

Random Forest:
Foram selecionadas as features 'preco', 'idCategory', 'sigla_uf', 'taxa_selic', 'taxa_exp_inflacao', 'USD' para o eixo X e ‘quantidade’ para o eixo Y

```
# Criar features e target
features = ['preco', 'idCategory', 'sigla_uf', 'taxa_selic', 'taxa_exp_inflacao', 'USD']
target = 'quantidade'

X = data[features]
y = data[target]
```

KNN: As features utilizadas nesse modelo foram ‘nu_cnpj_empresa’, ‘nu_processo’, ‘nu_registo_produto’, 'year_vencimento_registro' para o eixo x e “st_situacao_registro_valido” para o eixo y

```
X = df_dados_abertos_alimentos[['year_final_processo', 'nu_processo', 'nu_registro_produto', 'year_vencimento_registro']]
y = df_dados_abertos_alimentos[['st_situacao_registro_Válido']]
```

Gradient Boosting Regressor: No eixo x foram utilizadas as features "Preço", "CNPJ", "Valor", "Quantidade" e para o eixo y foi utilizada a ‘Encoded_labels’, obtida a partir da coluna ‘Produto’.

```
# Fit and transform the labels, then add a new column to the DataFrame
df_dados_da_api['Encoded_Labels'] = label_encoder.fit_transform(df_dados_da_api['Produto'])
```

```
coluna_alvo = 'Encoded_Labels'

# df_dados_caderneta[['renda_total']]

condicionantes_subgrupos = ["Preço", "CNPJ", "Valor", "Quantidade"]

X2 = df_dados_da_api[condicionantes_subgrupos]
y2 = df_dados_da_api[coluna_alvo]
```

4. Serviços utilizados

A arquitetura de big data oferece, entre seus principais benefícios, o processamento distribuído. Para explorar essa capacidade em consultas e análises de dados em larga escala, optou-se pelo uso da ferramenta PySpark.

Ao integrar o PySpark com o AWS Redshift, conseguimos aprimorar o processamento distribuído para análise de grandes volumes de dados. Essa integração capitaliza a eficiência do Redshift como um armazém de dados distribuído, aliada à capacidade do Spark. É crucial uma configuração meticulosa de parâmetros e uma atenção especial à segurança para assegurar um desempenho eficiente e proteger dados sensíveis ao longo do processo. Essa abordagem oferece uma solução versátil e escalável para análise de dados distribuída.

Exemplo de uso do Spark:

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Property used to format output tables better
spark
```

```
[ ] cnpj_vendas = spark.sql('''
    SELECT
        vendas.id,
        vendas.cnpj,
        vendas.idCategory,
        vendas.produto,
        vendas.data,
        vendas.preco,
        vendas.quantidade,
        tabela.sigla_uf
    FROM
        vendas
    INNER JOIN
        tabela ON vendas.cnpj = tabela.cnpj;
    ''')
```

```
[ ] cnpj_vendas_bacen = spark.sql('''
        SELECT
            cnpj_vendas.id,
            cnpj_vendas.cnpj,
            cnpj_vendas.idCategory,
            cnpj_vendas.produto,
            cnpj_vendas.data,
            cnpj_vendas.preco,
            cnpj_vendas.quantidade,
            cnpj_vendas.sigla_uf,
            bacen.USD,
            bacen.taxa_selic,
            bacen.taxa_exp_inflacao
        FROM
            cnpj_vendas
        INNER JOIN
            bacen ON cnpj_vendas.data = bacen.data;
    ''')
```

5. Modelagem

5.1 Escolha do Modelo Ensemble

Um modelo ensemble (conjunto) refere-se à combinação de múltiplos modelos de aprendizado de máquina para criar um modelo mais robusto e preciso do que os modelos individuais. A ideia subjacente aos modelos ensemble é que a combinação de várias fontes de informação pode superar as limitações de modelos individuais e produzir um desempenho geral mais forte.

A lista de modelos utilizados pelo grupo foram:

- **Gradient Boosting:** O modelo Gradient Boosting representa uma abordagem sofisticada e poderosa no campo do aprendizado de máquina, e sua escolha estratégica em nossa análise visa aproveitar sua capacidade única de construir modelos preditivos robustos por meio de um conjunto iterativo de modelos individuais de menor desempenho. Esta técnica opera de maneira sequencial e adaptativa, refinando continuamente suas previsões ao longo de múltiplas iterações.
  A essência do Gradient Boosting reside na correção dos erros residuais dos modelos anteriores. Inicialmente, um modelo simples é treinado para fazer previsões, e os erros são identificados. Em seguida, um segundo modelo é construído para corrigir esses erros específicos, e esse processo é repetido iterativamente. Cada novo modelo adicionado ao conjunto visa melhorar a precisão global do modelo, concentrando-se nas áreas onde as previsões anteriores foram menos precisas.
  O componente "Gradient" refere-se à utilização do gradiente da função de perda, o que significa que o modelo procura minimizar a função de perda em direção ao gradiente descendente. Isso proporciona uma abordagem eficiente para ajustar os parâmetros do modelo durante cada iteração, garantindo uma convergência mais rápida para uma solução otimizada.

- **Random Forest:** O Random Forest é um modelo de aprendizado de máquina pertencente à categoria de Ensemble Learning, comumente utilizado para tarefas de regressão. Enquanto o aprendizado de máquina tradicionalmente envolve a previsão de classes, a regressão com Random Forest visa prever valores numéricos. Durante o treinamento, várias árvores de decisão são construídas de forma aleatória e independente, cada uma utilizando um subconjunto aleatório dos dados e uma seleção aleatória de características. Durante a previsão, os resultados das árvores são combinados, geralmente através de uma média, para produzir uma estimativa mais robusta e precisa do valor desejado. O Random Forest para regressão destaca-se pela

sua capacidade de lidar com uma variedade de dados e pela reduzida propensão ao sobreajuste, proporcionando resultados mais confiáveis em comparação com modelos individuais de árvores de decisão.

5.2 Escolha de outros Modelos

Ampliando o espectro de análise no campo do aprendizado de máquina, exploramos modelos diversificados para identificar padrões em dados complexos. Além das tradicionais previsões unidimensionais, consideramos abordagens como clusterização, revelando grupos intrínsecos nos conjuntos de dados e permitindo adaptações específicas para diferentes categorias. A análise de séries temporais, por sua vez, oferece insights sobre a evolução temporal dos dados, antecipando padrões futuros. Ao integrar técnicas como clusterização e análise de séries temporais, obtemos uma compreensão mais holística e sofisticada, capacitando-nos a decifrar nuances complexas e aprimorar a identificação de padrões em dados multidimensionais.

- **Arima:** O modelo ARIMA (Autoregressive Integrated Moving Average) combina três componentes fundamentais: autoregressivo (AR), média móvel (MA) e diferenciação (I). O componente autoregressivo (AR) permite a modelagem da relação entre uma observação e seus valores anteriores, considerando a autocorrelação. A média móvel (MA), por sua vez, incorpora informações sobre os erros passados para capturar padrões de médias temporais. A diferenciação (I) é empregada para tornar a série temporal estacionária, facilitando a modelagem de tendências. Essa abordagem integrada fornece uma resposta robusta à complexidade dos padrões temporais, permitindo que o ARIMA se destaque na identificação, modelagem e previsão de tendências em dados temporais

- **KNN:** O KNN é um método de aprendizado supervisionado que se destaca pela sua simplicidade conceitual e eficácia em lidar com padrões complexos. Neste contexto, o KNN classifica ou realiza previsões para um ponto de dados com base na maioria das classes ou nos valores alvo dos K pontos mais próximos no espaço de características. A robustez do KNN reside na sua capacidade de capturar relações não lineares e considerar a estrutura local dos dados. Ao calcular a distância entre pontos, o modelo leva em conta as características relevantes para cada observação, permitindo uma adaptação dinâmica a padrões específicos em diferentes partes do espaço de características.

6. Avaliação

Com o propósito de realizar a construção do modelo preditivo possível, incorporamos diversas métricas para avaliação de desempenho, com ênfase em acurácia, Mean Squared Error e coeficiente de determinação (R2 Score). Também foi utilizada a matriz de correlação para verificar a eficiência do modelo

A acurácia assume um papel central ao proporcionar uma medida da precisão global do modelo. Essa métrica revela a proporção de previsões corretas em relação ao total de observações, destacando a habilidade do modelo em realizar previsões precisas de maneira abrangente.

O Mean Squared Error, por sua vez, desempenha um papel crucial ao quantificar a média dos quadrados dos erros entre as previsões do modelo e os valores reais. Ao penalizar mais significativamente os erros de maior magnitude, o MSE fornece uma indicação clara da dispersão dos erros, indicando se o modelo está produzindo previsões próximas aos valores reais.

O coeficiente de determinação (R2 Score) oferece uma análise da proporção da variabilidade nos dados de saída que é explicada pelo modelo. Variando de 0 a 1, um R2 próximo de 1 sugere um ajuste ideal, indicando que o modelo está capturando efetivamente a variabilidade dos dados. Por outro lado, valores mais baixos sugerem que o modelo pode não estar capturando completamente essa variabilidade.

Também foi utilizado a “feature importance” que avaliou a relevância das características (features) no modelo. Essa análise destaca quais atributos têm maior impacto na predição, contribuindo para a compreensão dos fatores mais influentes.

6.1 Avaliação de Desempenho

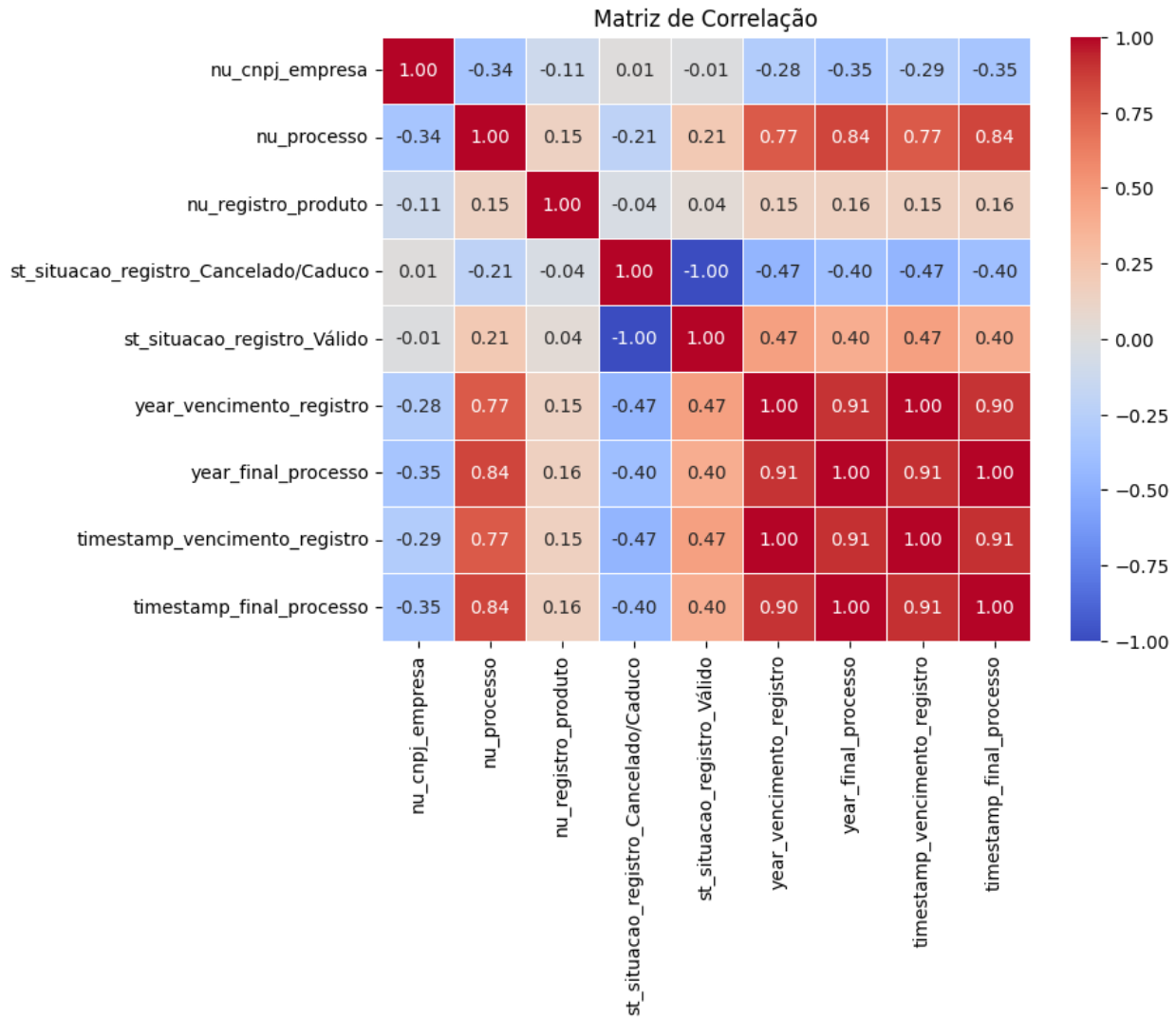
Para determinar a utilização dos modelos e com base nas premissas que foram tomadas, plotamos alguns gráficos e visualizamos as métricas para embasar o cruzamento dessas informações com a intenção de obter os melhores resultados possíveis.

Segue as avaliações a partir dos modelos que rodamos:

KNN - Dados Abertos da Anvisa

Tabelas: Dados Abertos de Alimentos

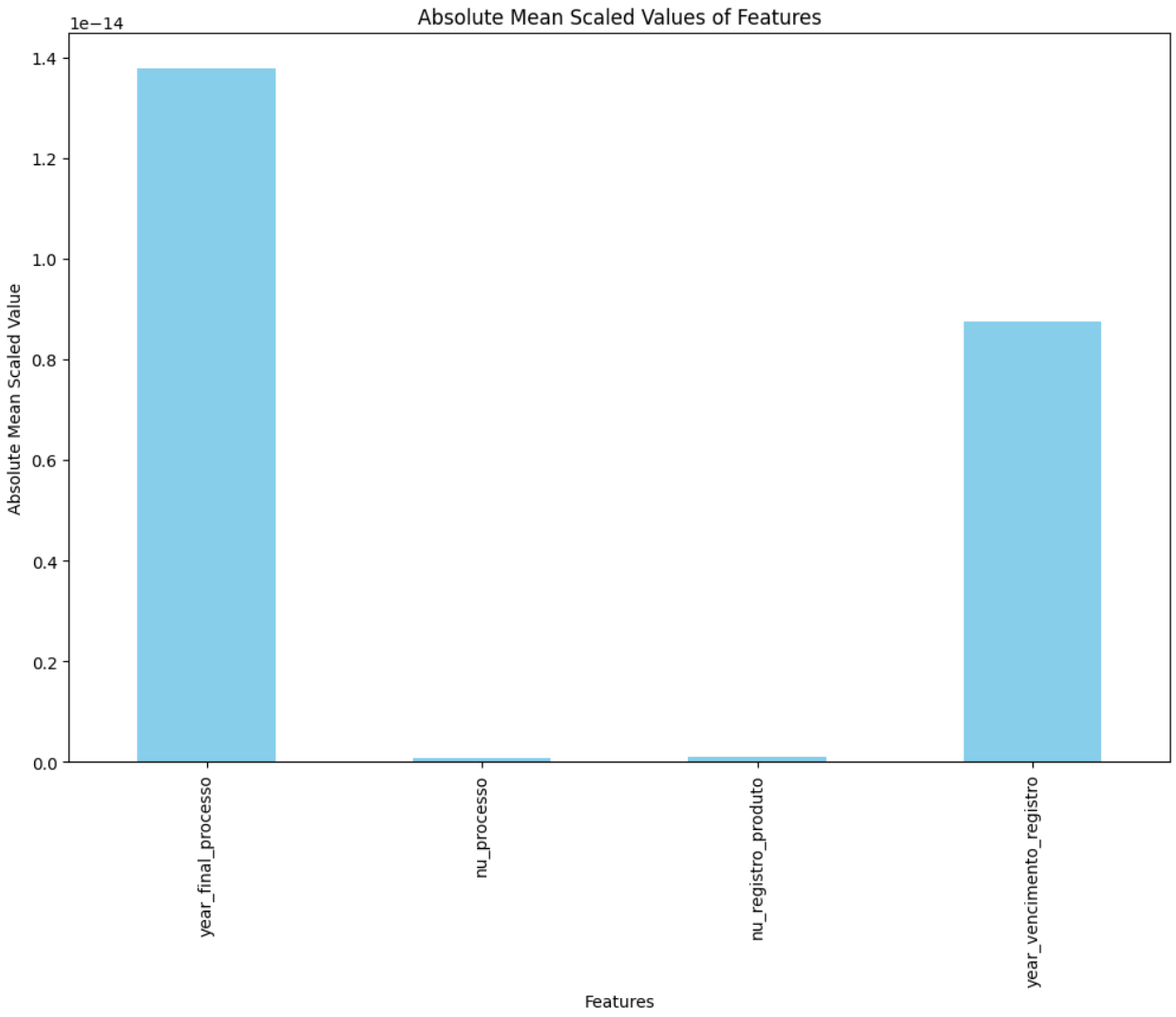
A partir das análises iniciais realizadas, visualizamos que a matriz de correlação denota a relação de algumas variáveis importantes para pensarmos na utilização em si. A matriz é codificada por cores, com vermelho indicando uma correlação positiva e azul indicando uma correlação negativa. Quanto mais escuro for a cor, mais forte será a correlação. A partir desta matriz, podemos ver que há uma forte correlação positiva entre “nu_cnpj_empresa” e “nu_processo”, e uma forte correlação negativa entre “nu_cnpj_empresa” e “timestamp_final_processo”. Isso sugere que empresas com um número maior de processos tendem a ter um número maior de processos finalizados, e empresas com um número menor de processos tendem a ter um número menor de processos validados. Segue matriz de correlação:



Outra métrica de desempenho das features que utilizamos foi a partir dos valores médios absolutos “scaled” das características do modelo. As características são “ano de finalização do processo”, “número do processo”, “número de registro do processo” e “ano de vencimento do registro”. A característica “ano de finalização do processo” tem o valor mais alto, se destacando como uma feature muito relevante para o modelo, seguida pelo “ano de vencimento do registro”. Já as features, “número do processo” e “número de registro do processo” têm valores mais baixos. Isso sugere que as primeiras duas características têm uma relevância maior para consideração.

A importância das características em um modelo pode ser avaliada de várias maneiras. Uma delas é o cálculo da importância relativa de cada característica. Existem várias técnicas para calcular a importância das características, como árvores de decisão, florestas aleatórias, modelos lineares e redes neurais. Uma técnica comum para calcular a importância das características é o cálculo do coeficiente de correlação entre cada característica e a variável de saída. O coeficiente de correlação mede a força da relação linear entre duas variáveis. Se duas variáveis têm uma correlação positiva, significa que o aumento de uma é acompanhado pelo aumento da outra. Já uma correlação negativa significa que a diminuição de uma é acompanhada pelo aumento da outra e vice-versa. Quando uma variável não apresenta uma correlação linear com outra variável, a correlação é zero. Isso geralmente acontece quando são observados dois eventos aleatórios independentes.

Tabela de features mais importantes pro modelo:



Nos atentamos, também, a outras métricas adicionais para cálculo do erro das previsões com base nas features selecionadas:

Os resultados das métricas de avaliação fornecem insights cruciais sobre o desempenho do modelo em questão. Vamos analisar cada métrica individualmente:

Mean Squared Error (MSE): 0.0453 → O MSE representa a média dos quadrados dos erros entre as previsões do modelo e os valores reais. Neste contexto, o valor de 0.0453 indica uma baixa dispersão dos erros, sugerindo que o modelo tem uma capacidade razoável de fazer previsões precisas.

R2 Score: -0.1821 → O R2 Score, também conhecido como coeficiente de determinação, mede a proporção da variância na variável dependente que é previsível a partir das variáveis

independentes. Um valor negativo sugere que o modelo não está performando bem em relação a uma linha horizontal básica. Idealmente, desejaríamos um R2 Score mais próximo de 1 para indicar uma boa adequação do modelo aos dados.

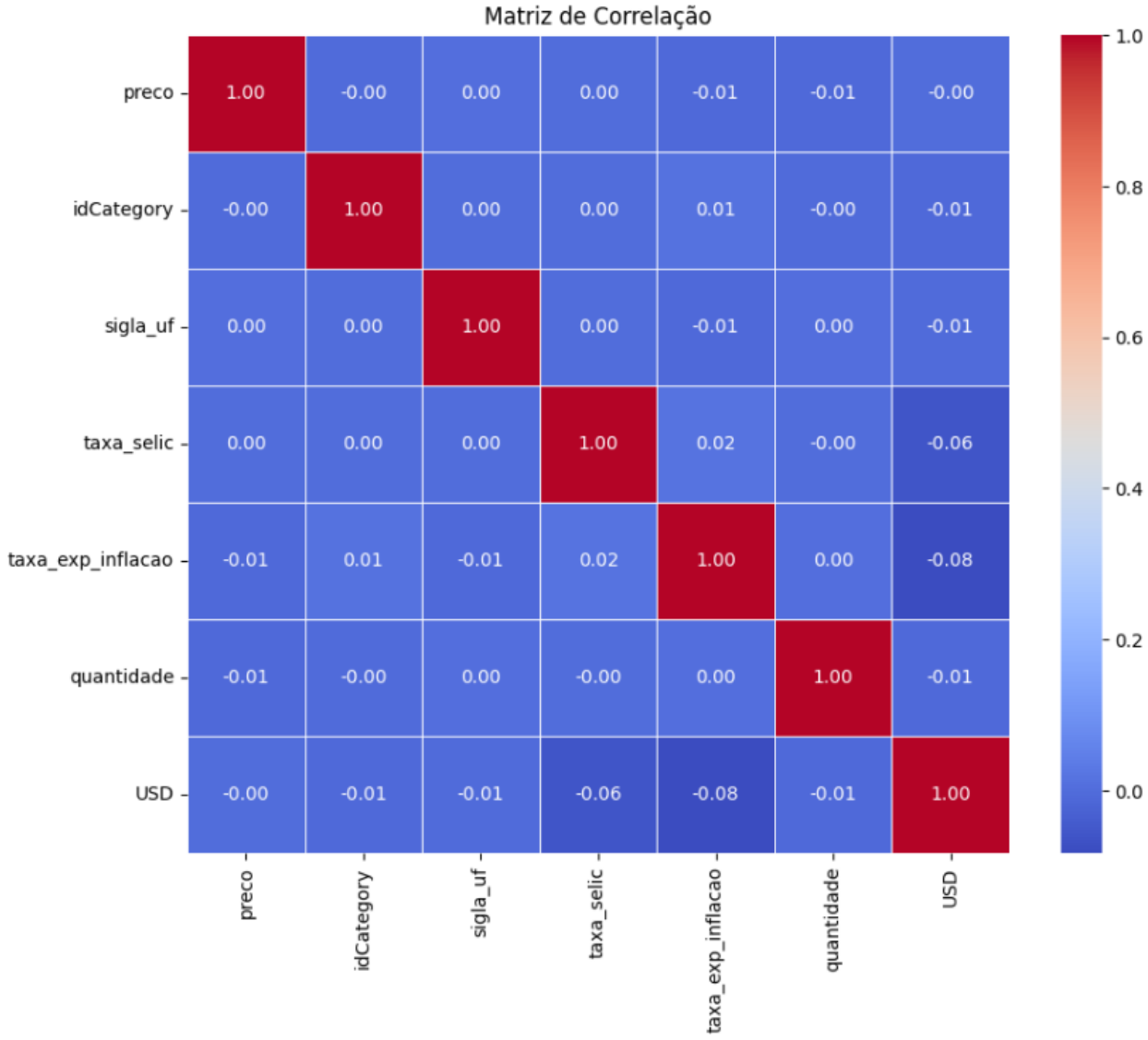
Mean Absolute Error (MAE): 8.5935 → O MAE representa a média absoluta dos erros entre as previsões e os valores reais. Um valor de 8.5935 indica a magnitude média dos erros cometidos pelo modelo. Quanto mais próximo de zero, melhor o desempenho, indicando menor discrepância entre as previsões e os valores reais.

Root Mean Squared Error (RMSE): 0.2129 → O RMSE é a raiz quadrada do MSE e fornece uma métrica da média dos erros de previsão, penalizando erros maiores. Um valor de 0.2129 sugere uma dispersão relativamente baixa dos erros, indicando uma precisão razoável nas previsões do modelo.

Random Forest - Junção de Dados com base nas Vendas (API)
Tabela: Junção das tabelas: Sales(API do Cliente), Concatenação das 4 tabelas CNPJ e Bacen
Target: Prever a demanda (quantidade)

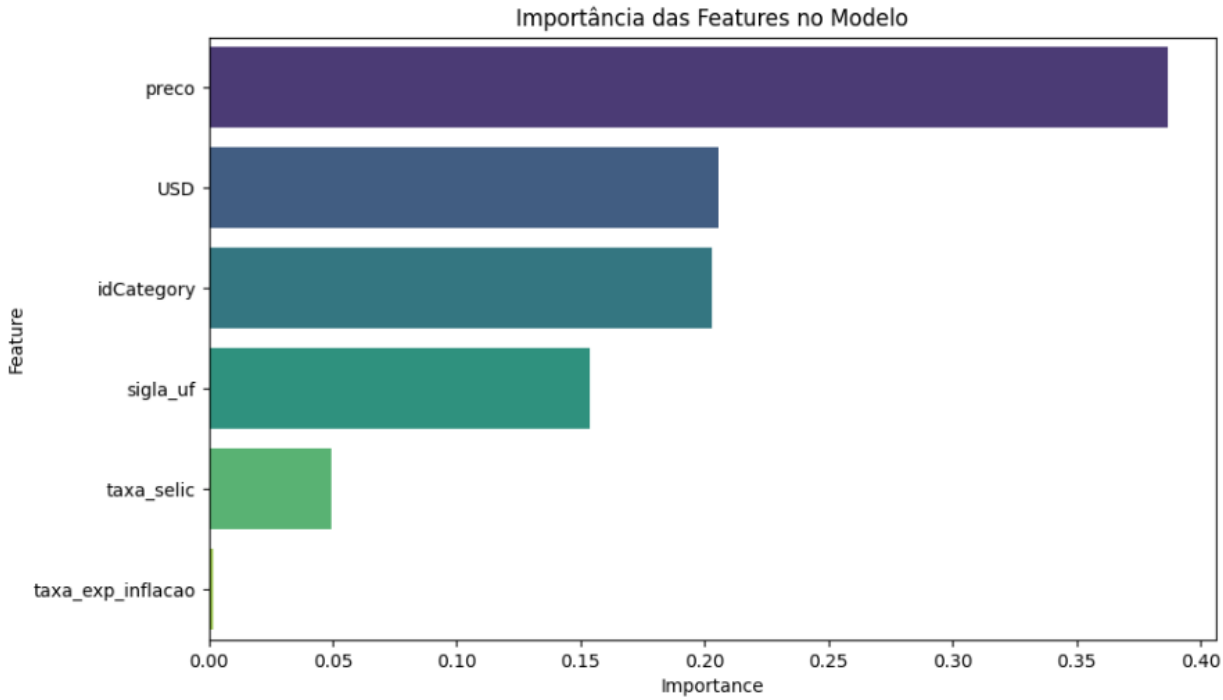
Durante a análise exploratória, examinamos os coeficientes associados a diferentes variáveis em nosso conjunto de dados. Notamos que o coeficiente relacionado ao preço é muito próximo de zero, indicando uma contribuição insignificante para a variação na quantidade vendida. Por outro lado, a variável "Quantidade" apresenta um coeficiente próximo de 1, o que era esperado, pois é a variável que buscamos prever. Ao avaliar a "Categoria do Produto" (idCategory),

observamos um coeficiente próximo de zero, sugerindo um impacto limitado na quantidade vendida. A "Sigla UF" (sigla_uf) apresenta um coeficiente zero, indicando que essa variável não contribui para a variação na quantidade vendida, possivelmente devido à codificação ou à falta de variação nos dados. Além disso, variáveis como "Taxa Selic," "Taxa de Exportação Inflação," e "Dólar (USD)" exibem coeficientes próximos de zero, indicando contribuições mínimas para a variação na quantidade vendida. A conclusão retirada destas observações é que, dado o caráter fictício dos dados de venda, é compreensível a ausência de relações substanciais entre as features analisadas.



Ao expandir nossa análise para incluir a avaliação do modelo de regressão Random Forest, exploramos métricas essenciais que oferecem insights sobre o desempenho e a precisão de nossas previsões. **O Erro Quadrático Médio (Mean Squared Error)** foi calculado em **888.5589**, indicando a média dos quadrados dos erros entre as previsões e os valores reais. Paralelamente, **o Erro Médio Absoluto (Mean Absolute Error**) foi registrado em **25.5177**,

representando a média das diferenças absolutas entre previsões e observações reais. **O Erro Quadrático Médio Raiz (Root Mean Squared Error)** foi calculado como **29.8087**, proporcionando uma visão mais intuitiva da dispersão dos erros. Contudo, a métrica-chave, o coeficiente de determinação R^2, revelou um valor muito próximo de zero, sugerindo que o modelo pode não estar capturando efetivamente os padrões nos dados, indicando a necessidade de refinamento e aprimoramento. **O Coeficiente de Correlação de Pearson**, que mede a relação linear entre as variáveis, foi estimado em **0.0026**, indicando a ausência de uma relação forte entre as variáveis analisadas. Esses resultados apontam para a complexidade na modelagem dos padrões subjacentes aos dados e destacam a importância de aprimorar nosso entendimento das interações entre as variáveis para fortalecer as capacidades preditivas do modelo.



Também exploramos a importância das features no model. Não surpreendentemente, a característica "preco" emerge como a mais significativa, desempenhando um papel central na modelagem e previsão da quantidade vendida. Em segundo lugar em termos de importância, o "Dólar (USD)" demonstra sua influência substancial, destacando-se como uma variável crucial na compreensão das dinâmicas de vendas."idCategory", "sigla_uf" e "taxa_selic" apresentam importância semelhante, indicando que, embora possam desempenhar papéis distintos, sua contribuição para a variação na quantidade vendida é comparável. No entanto, a constatação é que a variável "taxa_selic" não demonstra relevância ao modelo, sugerindo que, na configuração atual, ela não oferece insights significativos para a predição da quantidade vendida. Essa compreensão mais granular da importância relativa das features destaca áreas específicas para foco e refinamento, proporcionando direcionamento valioso para futuras iterações do modelo e aprimoramento de sua capacidade preditiva.

7. Otimização do modelo

Aprimorar a eficiência do modelo preditivo é necessário para a obtenção de resultados mais precisos e ágeis. Nessa busca por aprimoramento, diversas estratégias de otimização podem ser empregadas. Estas estratégias não apenas refinam a capacidade preditiva do modelo, mas também potencializam sua eficácia em lidar com conjuntos de dados complexos.

Abaixo estão as técnicas utilizadas.

7.1 Utilização de Técnicas para Otimização

7.1.2 Hiperparâmetros:

A hiperparametrização é o processo de ajuste de configurações externas ao modelo de aprendizado de máquina para otimizar seu desempenho. Envolve a escolha inicial de hiperparâmetros, definição de métricas de avaliação, divisão de dados em conjuntos de treinamento, validação e teste, treinamento do modelo, ajuste iterativo de hiperparâmetros com base nos resultados da validação, e avaliação final no conjunto de testes. Métodos automatizados, como pesquisa em grade ou otimização bayesiana, são comumente usados para tornar esse processo eficiente.Exemplos de hiperparâmetros:

Em Random Forest:

- Número de Árvores (n_estimators):Este parâmetro define quantas árvores de decisão compõem o Random Forest. Aumentar o número de árvores geralmente leva a um modelo mais robusto, mas há um ponto de diminuição dos retornos.

- Profundidade Máxima das Árvores (max_depth):Controla a profundidade máxima de cada árvore de decisão. Ajustar este parâmetro pode ajudar a evitar o sobreajuste (overfitting) ou sub ajuste (underfitting) do modelo.

- Número Mínimo de Amostras para Divisão (min_samples_split):Define o número mínimo de amostras necessárias para uma divisão em um nó. Ajustar este parâmetro pode influenciar a complexidade das árvores.

- Número Mínimo de Amostras em Folhas (min_samples_leaf):Especifica o número mínimo de amostras permitidas em uma folha. Isso afeta a granularidade das decisões tomadas por cada árvore.

- Máximo de Características Consideradas (max_features):Controla o número máximo de características consideradas para fazer uma divisão em cada nó. Isso pode influenciar a diversidade entre as árvores

- Bootstrap (bootstrap):Indica se deve ser usado o método de bootstrap (amostragem com substituição) ao construir as árvores. Isso pode afetar a variabilidade entre as árvores.

Além disso, é possível utilizar técnicas de validação cruzada, como o k-fold cross-validation, para avaliar o desempenho do modelo em diferentes conjuntos de dados.

É importante ressaltar que neste processo inicial, nenhum dos hiperparâmetros foram utilizados a modo de melhorar os resultados obtidos após avaliação realizada.

8. Resultados obtidos

Considerando a otimização da visualização dos dados para o parceiro, compreender a correlação entre diferentes tabelas é muito importante para criar infográficos mais informativos. Os resultados obtidos foram interessantes com base nos joins entre tabelas, permitindo análises não previamente pensadas. A partir da matriz de correlação, identificamos as colunas com maior "afinidade" entre si, proporcionando insights valiosos para a estruturação eficiente dos dados.

8.1.1 KNN - Tabelas de Dados da Anvisa (Dados Abertos de Alimentos)

O processo de construção e avaliação do modelo ensemble utilizando o algoritmo k-Nearest Neighbors (KNN) nos ofereceu alguns insights valiosos sobre seu desempenho e capacidade preditiva. Inicialmente, a escolha criteriosa das variáveis independentes, como 'year_final_processo', 'nu_processo', 'nu_registro_produto', e 'year_vencimento_registro', nos forneceu informações relevantes ao modelo.

Após a preparação dos dados e a divisão em conjuntos de treinamento e teste, a aplicação do KNN com 5 vizinhos revela uma acurácia geral de 95%, sugerindo uma capacidade positiva de previsão. No entanto, uma análise mais detalhada por meio da matriz de confusão revela um desequilíbrio nas previsões, com uma tendência do modelo em classificar predominantemente a classe majoritária (0). Isso resulta em verdadeiros negativos e falsos positivos em números significativos, enquanto os verdadeiros positivos e falsos negativos para a classe minoritária (1) são relativamente baixos.

O relatório de classificação confirma essa observação, destacando uma precisão notavelmente baixa para a classe 1 (st_situacao_registro_Válido), indicando dificuldade do modelo em identificar corretamente os casos positivos. As métricas de recall e f1-score também são comprometidas para a classe 1, apontando para uma performance limitada nessa identificação.

Essa limitação é atribuída ao desequilíbrio no número de amostras entre as classes, onde a classe 0 é majoritária.

As métricas de erro, incluindo Mean Squared Error (MSE), R2 Score, Mean Absolute Error (MAE) e Root Mean Squared Error (RMSE), proporcionam uma visão adicional da qualidade das previsões do modelo. O baixo MSE sugere concordância entre previsões e valores reais, mas o R2 Score negativo indica que o modelo não supera um modelo ingênuo. O MAE e RMSE fornecem uma compreensão da magnitude e dispersão dos erros, destacando a importância de considerar o desafio de lidar com classes desbalanceadas.

8.1.2 Random Forest - Tabelas de Vendas + Dados do Bacen + CNPJ (join)

A avaliação do modelo com base nas tabela que fizemos, nos forneceu alguns insights valiosos sobre a contribuição relativa das diferentes features para as previsões. Inicialmente, é importante ressaltar que o Random Forest é uma ferramenta poderosa para tarefas de regressão, destacando-se pela capacidade de lidar com diversos tipos de dados e pela mitigação do sobreajuste comumente associado a modelos de árvores de decisão individuais.

Os resultados das métricas de erro, incluindo o Erro Quadrático Médio (888.56), o Erro Médio Absoluto (25.52), e o Erro Quadrático Médio Raiz (29.81), oferecem uma visão quantitativa da precisão das previsões. No entanto, o coeficiente de determinação R^2 próximo de zero sugere que o modelo pode não estar efetivamente capturando os padrões nos dados, levantando questões sobre sua capacidade de explicar a variação na quantidade vendida.

A análise das importâncias das features destaca a relevância diferencial de cada variável para as previsões do modelo. 'Preco' emerge como a característica mais importante, seguida por 'USD'. Intrigantemente, 'idCategory', 'sigla_uf', e 'taxa_selic' apresentam a mesma relevância, enquanto 'taxa_selic' não demonstra qualquer contribuição significativa para as previsões.
Ao analisar cada feature individualmente, observamos que 'preco' tem um coeficiente próximo de zero, indicando uma contribuição insignificante para a variação na quantidade vendida. Por outro lado, 'quantidade' tem um coeficiente próximo de 1, como esperado, sendo a variável alvo.

As conclusões finais ressaltam a natureza fictícia dos dados de venda, explicando a ausência de relações substanciais entre as features. Isso destaca a importância de considerar o contexto dos dados ao interpretar os resultados do modelo. A conclusão final reforça a necessidade de uma abordagem crítica ao avaliar modelos em cenários onde a relação entre variáveis pode não seguir padrões convencionais, como é o caso dos dados fictícios de venda apresentados.
Ao avaliar a aplicação na API do cliente, concluímos que, devido à natureza fictícia dos dados, é impossível para um modelo de IA obter bons resultados. A falta de relações (coeficientes) significativas entre as informações torna-as incompatíveis com o mundo real. No entanto,

ressaltamos que, em um ambiente de produção com dados autênticos, o modelo pode alcançar resultados superiores, contribuindo para análises mais precisas e decisões informadas.