UNIVERSIDADE FEDERAL DE SERGIPE
DEPARTAMENTO DE COMPUTAÇÃO

Aprendizagem de Máquina
Hendrik Teixeira Macedo

Implantação de algoritmo de aprendizagem de máquina para previsão de vendas em série temporal

Carlos Daniel Lima de Gois
João Pedro Cardoso Arruda
Rafael Nascimento Andrade

SÃO CRISTÓVÃO

20/02/2026

1. Introdução

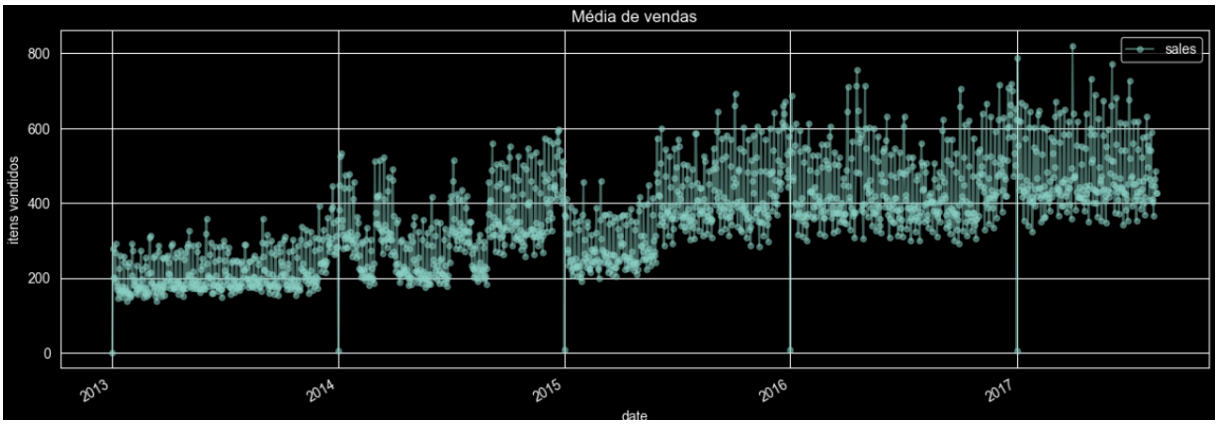
Esse documento se trata de um relatório técnico sobre a realização de um trabalho da disciplina de Aprendizagem de Máquina da Universidade Federal de Sergipe. O qual consiste na participação em uma competição do Kaggle que envolve a implementação de um algoritmo de aprendizagem de máquina para previsões. A competição em questão é a "Store Sales - Time Series Forecasting" e consiste em implementar um modelo para prever o número de vendas de determinados produtos em determinadas lojas no Equador dentro dos dias especificados em uma série temporal. O trabalho foi feito em um notebook python utilizando as bibliotecas numpy, pandas, scikit learn, matplotlib e xg boost.

2. Análise Exploratória dos Dados

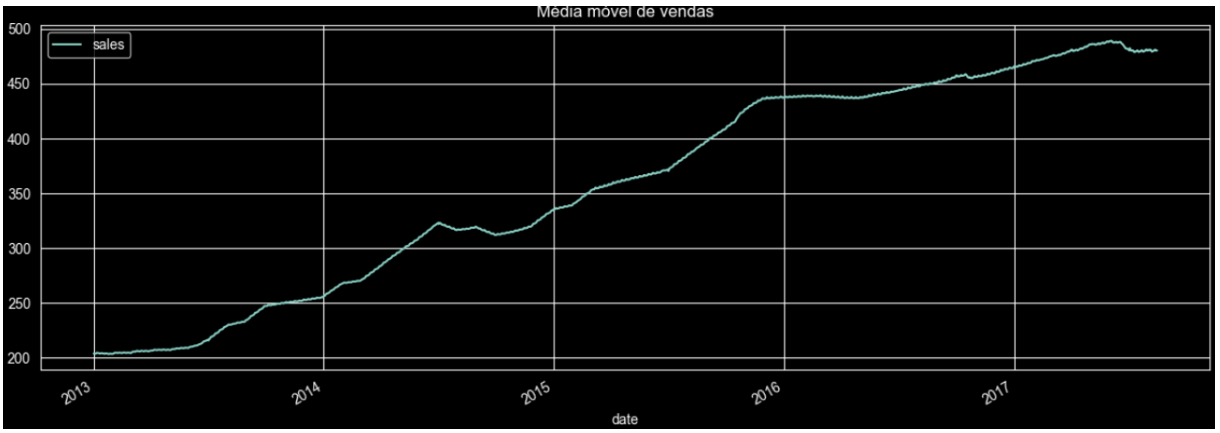
Os Dados fornecidos pela competição são alguns arquivos de extensão .csv, são eles:

- **train.csv:**
  Arquivo com os dados de treino com registros contendo o total de vendas de uma família de produtos em uma loja específica em um determinado dia, a tabela já vem ordenada por data em ordem crescente, possui registros de dias consecutivos sem lacunas e contém as seguintes colunas:
  - **id:** Simplesmente um identificador único de cada linha.
  - **date:** Datas, mais especificamente os dias das vendas.
  - **store_nbr:** Identificador da loja onde foram feitas as vendas, com um total de 54 lojas diferentes.
  - **family:** Família dos produtos vendidos, com um total de 33 famílias diferentes.
  - **sales:** Quantidade de itens vendidos daquela família naquela loja naquele dia.
  - **onpromotion:** Quantidade de itens daquela família que estavam em promoção naquela loja.
- **stores.csv:**
  Informações sobre as lojas incluindo a cidade, o estado, o tipo e um coluna de cluster para indicar grupos de lojas similares
- **oil.csv:**
  Informações sobre os preços do petróleo para cada dia, porém alguns dias não possuem preço definido, possivelmente porque não houveram negociações de petróleo naquele dia, além disso alguns dias apareciam com múltiplos registros de preços, possivelmente devido a oscilações no mercado.
- **holiday_events.csv:**
  Informações sobre os dias de feriados do Equador, incluindo a data, o tipo, a abrangência (Local, Regional ou Nacional), nome, descrição e uma indicação de se o feriado foi transferido de data. Adicionalmente as datas resultantes das transferências também aparecem como registros.
- **test.csv:**
  Os dados de teste nos quais o modelo deve fazer a previsão em cima para submeter a resposta. Possui as mesmas colunas do train.csv exceto pela ‘sales’. Também vem ordenado por data em ordem crescente e as datas começam onde as de train.csv terminam, caracterizando uma continuação dentro de uma série temporal.

Para ter uma visualização melhor sobre a tendência geral das vendas ao longo dos dias decidimos analisar a média de vendas diárias de todos os produtos em todas as lojas.

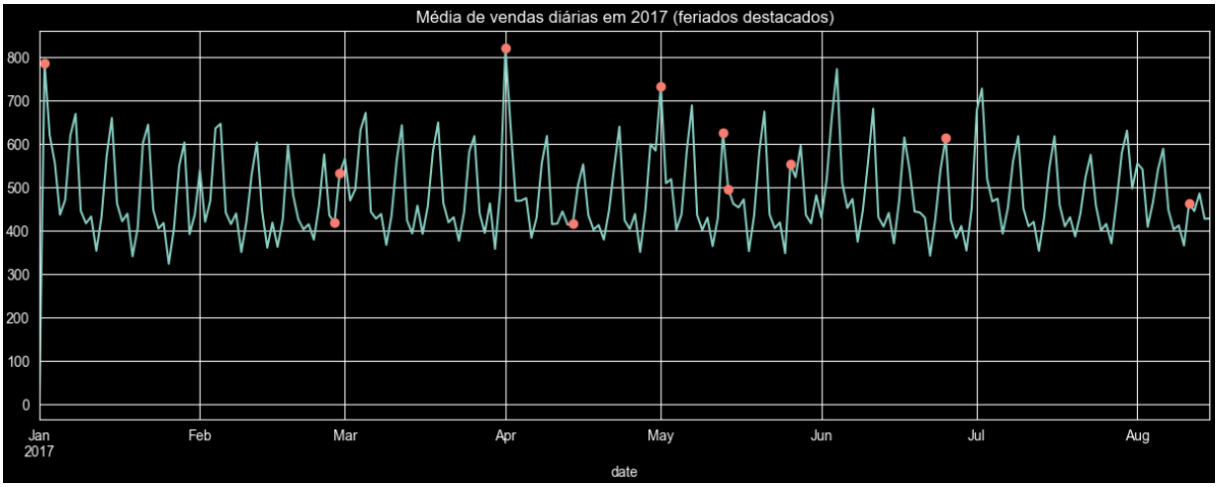


Ao observar o gráfico houve a suspeita da existência de uma tendência de aumento das vendas ao longo do tempo. Além disso, foram percebidas oscilações frequentes que poderiam indicar a presença de ciclos ou de sazonalidade nas vendas. Para tentar identificar a tendência decidimos fazer um gráfico com a média móvel das vendas considerando uma janela de um ano para suavizar as oscilações.



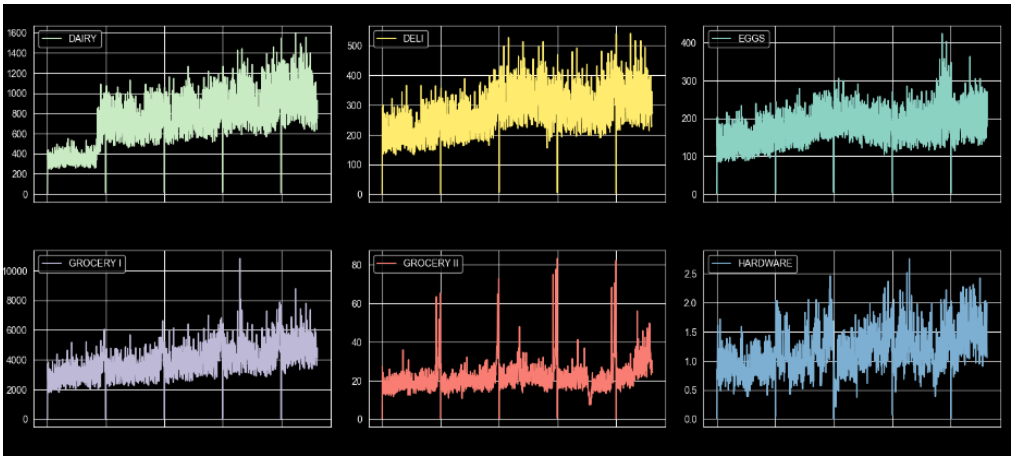
Ao analisar o gráfico da média móvel foi possível concluir que de fato há uma tendência de aumento nas vendas ao longo do tempo.

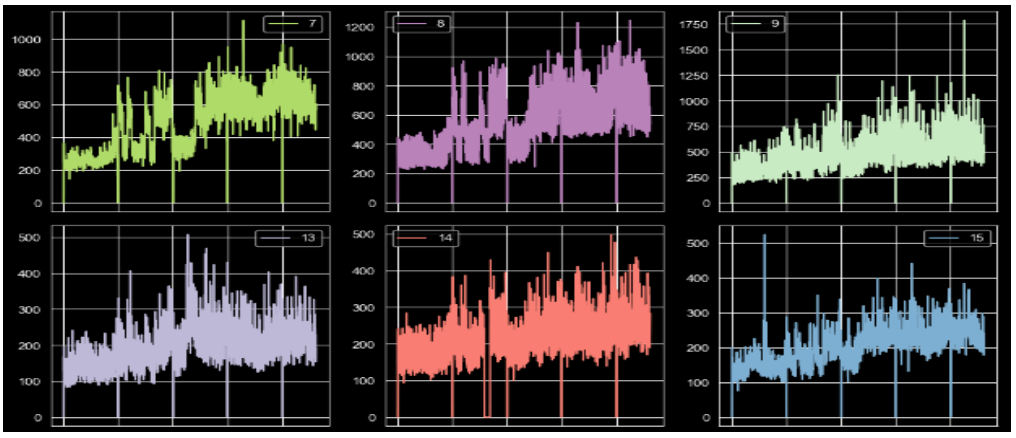
Para tentar identificar com mais clareza se há a presença de oscilações foi selecionado um período mais curto do gráfico com a média de vendas diárias e realçadas as datas em que houveram feriados regionais e nacionais.



Ao analisar esse gráfico que possui formato de onda foi possível concluir que de fato há a presença de ciclos nas vendas, além disso, foi possível perceber que houveram alguns picos mais elevados nos dias de alguns dos feriados.

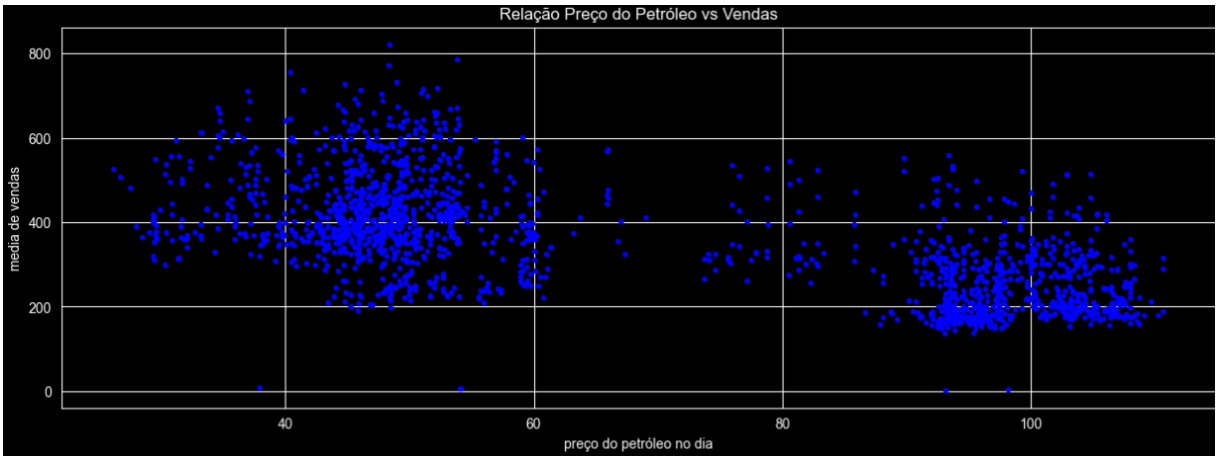
Para não generalizar tanto nossa análise decidimos observar os gráficos com as médias de vendas diárias para cada loja e também para cada família de produtos.



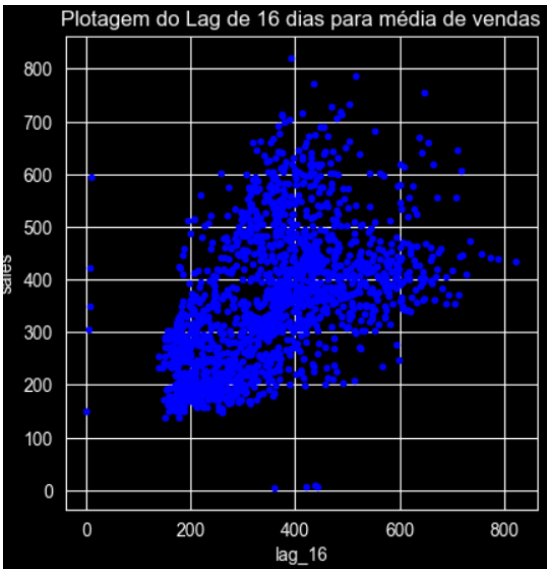


No fim foi observado que a maioria desses gráficos possuem comportamento similar ao gráfico de média de vendas diárias geral, porém cada um com suas variações, o que indica que além da data, a família do produto e a loja também influenciam no número de vendas.

Observamos também a relação do preço do petróleo com o número de vendas e foi possível perceber que há uma influência, de modo que a média de vendas tende a diminuir com o aumento do preço do petróleo.



Finalmente, por termos identificado ciclos e uma tendência, decidimos verificar se havia influência dos números de vendas dos dias anteriores nas vendas seguintes e observamos que havia uma pequena relação de colinearidade.



Foi escolhida a observação de 16 dias antes pois a última data dos dados de teste é justamente 16 dias após a última data dos dados de treino e com isso para intervalos menores alguns registros do dataset de teste ficariam com essa informação faltando.

3. Escolha das Features

Após analisar os dados fomos capazes de escolher e criar algumas features, divididas em:

- Features de categorias
  Foram selecionadas algumas features que podem ser classificadas em um número finito de categorias. entre elas:
  - Features da loja:
    Features que indicavam informações a respeito da loja, pois foi constatado que essa tinha influência nas vendas a partir do gráfico, elas são:
    - identificador da loja (store_nbr)
    - cidade da loja
    - estado da loja
    - tipo da loja
    - cluster da loja
  - Features dos feriados:
    Features com informações a respeito do feriado pois foi constatado que eles podem gerar picos de vendas, elas são:
    - tipo do feriado
    - abrangência
    - localização
    - descrição
  - Família do produto (family)
- Features de venda
  Feature relacionadas a preços e vendas
  - número de produtos da família em promoção (onpromotion)
  - preço do petróleo
- Features de data
  Features relacionadas a data das vendas, utilizadas para tentar capturar as tendências temporais junto com a sazonalidade
  - time dummy: indicador de quantos dias desde o primeiro dia registrado no dataset de treino, usado para ajudar na modelagem da tendência temporal
  - mês
  - dia do mês
  - dia da semana
  - dia do ano
  - se é final de semana
  - se é dia de pagamento: no Equador os funcionários recebem seu salário no dia 15 ou no final do mês
- Features de Fourier
  Features representadas por funções de seno e cosseno para tentar modelar as sazonalidades relacionadas às datas e tentar identificar os ciclos presentes dentro das semanas, dos meses e dos anos.

- seno dia da semana: $sen(\frac{2\pi}{7} \times diaDaSemana)$
- cosseno dia da semana: $cos(\frac{2\pi}{7} \times diaDaSemana)$
- seno dia do mês: $sen(\frac{2\pi}{numeroDeDiasNoMes} \times diaDoMes)$
- cosseno dia do mês: $cos(\frac{2\pi}{numeroDeDiasNoMes} \times diaDoMes)$
- seno mês do ano: $sen(\frac{2\pi}{12} \times mesDoAno)$
- cosseno mês do ano: $cos(\frac{2\pi}{12} \times mesDoAno)$
- seno dia do ano: $sen(\frac{2\pi}{numeroDeDiasNoAno} \times diaDoAno)$
- cosseno dia do ano: $cos(\frac{2\pi}{numeroDeDiasNoAno} \times diaDoAno)$

- Features de atraso (Lag)
  Features representando as observações de vendas passadas, foi selecionado o atraso (lag) de 16 dias, que representa as vendas daquela família de produtos naquela loja 16 dias atrás. A escolha do intervalo de 16 dias foi porque se fosse escolhido um intervalo menor alguns registros do dataset de teste ficariam com esse valor faltando pois nos dados de teste a data vai até 16 dias depois da última data do dataset de treino. Além disso, na tentativa de observar as tendências foram utilizadas médias móveis no lag como features para capturar a tendência sem as oscilações das última semana, último mês e último ano. No total as features foram:
  - lag de 16 dias
  - média móvel de 7 dias para o lag de 16 dias
  - média móvel de 30 dias para o lag de 16 dias
  - média móvel de 365 dias para o lag de 16 dias

4. Pré-processamento dos dados

Para o pré-processamento de dados foi utilizada a biblioteca pandas do python, as tabelas dos arquivos .csv foram lidas e armazenadas em DataFrames.

Para lidar com os valores faltantes de preço do petróleo foi usado o .ffill() que implementa o método forward fill e preenche os valores faltantes com o último valor observado, exceto para a o valor da primeira data que foi preenchido com .bfill() o backward fill que preenche com a próxima observação. Já para os dias com múltiplos registros de preços do petróleo foi aplicada a média desses preços para o dia.

Na tabela de feriados os feriados registrados como transferidos foram removidos, pois estes não caem na data oficial e portanto não tem efeito de feriado.

As tabelas foram mergeadas para uma tabela só para gerar as features de entrada do modelo e nos dias em que não eram feriado a coluna do tipo de feriado (holiday_type) foi preenchida com valor ‘Work Day’.

As features de categorias com valores de string foram tratadas com o LabelEncoder do scikit learn para transformá-las em valores numéricos, visto que muitos modelos não aceitam o valor em string.

Para gerar os valores das features para o dataset de teste esse foi concatenado com o dataset de treino antes dos merges, para diferenciar os registros de cada um foi usada uma coluna temporária ‘is_train’, após a geração das features elas foram separadas novamente para o treino do modelo.

5. Seleção do Modelo e Treinamento

Devido ao seu histórico de vitórias em competições do Kaggle e pelo fato de que boa parte das features eram de categorias, o modelo escolhido foi o XGBoost.
Antes do treino separamos os dados do dataset de treino em 70% de dados para treinar o modelo e 30% dos dados para validação, como os dados se tratavam de uma série temporal separamos por data e mantivemos datas consecutivas dentro de cada conjunto, a divisão foi a seguinte, registros de antes da data de divisão ficaram no conjunto de treino e os dessa data em diante ficaram no conjunto de validação.
A métrica de erro utilizada foi a do RMSLE (Root Mean Squared Logarithmic Error), a raiz do erro quadrático médio logarítmico, que foi a métrica de erro estabelecida pela própria competição, a fórmula dela é a seguinte:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(1 - \hat{y}_i) - log(1 - y_i))^2}$$

Após treinar o modelo pela primeira vez, fizemos as medições dos erros tomando como referência o benchmark estabelecido pela própria competição que foi de 4.37970. O erro apresentado no conjunto de treino foi de 4.0152788162231445, um pouco melhor do que o benchmark, porém o erro observado no conjunto de validação foi de 3.55206036567688, que é melhor que o erro do treino o que indica que não houve overfitting do modelo. Uma possível explicação para isso pode ser que no conjunto de treino haviam muitos valores faltantes relacionados às features de atraso (lags), uma vez que no conjunto de treino estavam as primeiras datas, que não poderiam possuir valores de 16, 30 ou 365 dias atrás, como no caso das médias móveis de lags também, então talvez por isso os dados de treino fossem mais difíceis de prever.
Finalmente após o treino e a validação foi aplicada a previsão do modelo nos dados de teste e foi feita a submissão no Kaggle. Após submeter a resposta verificamos que o erro do modelo, informado pelo próprio Kaggle, no conjunto de teste foi de 2.78023, ainda melhor do que o erro de validação e consequentemente do de treino também confirmando que não houve overfitting.
Contudo, no intuito de melhorar a performance do modelo fizemos mais um tratamento nos dados, tomando como motivação o alto erro nos exemplos de treino em relação aos de validação e teste e então decidimos remover do conjunto de teste os registros que ficaram com valores faltantes para as features de lag. Com isso tivemos uma melhora significativa em relação ao primeiro treino, pois o novo erro nos exemplos de treino foi de 1.3896397352218628 e nos de validação foi de 1.2775474786758423. Com esse resultado podemos de novo confirmar que não houve overfitting, pelo mesmo motivo do primeiro caso de treino, já que o erro no

conjunto de validação foi similar mas levemente melhor do que no conjunto de treino.

6. Resultado Final

Após finalizarmos o treino do modelo submetemos a segunda previsão na plataforma do Kaggle, o resultado foi um erro de 1.25401 em cima dos exemplos de teste, um resultado melhor do que a primeira submissão, confirmando que o segundo treino foi mais eficaz que o primeiro e acima de tudo um erro próximo ao dos exemplos de treino, nos dando a confirmação de que não houve overfitting. Por fim esse erro também foi muito melhor do que o benchmark de 4.37970 definido pela competição, o que nos fez concluir que o modelo teve um desempenho satisfatório.