Relatório de Data Mining

Trabalho de Conclusão de Disciplina

Maio/2021

Professora Manoela Kohler

**Previsão de vendas para uma empresa varejista de calçados**

Participantes:

* Gustavo Dutra Martins
* Alex Marques Campos
* Fernanda

O arquivo jupyter utilizado em nosso projeto pode ser acessado no Github através do endereço:

<https://github.com/gdutramartins/bi-master/blob/main/Trabalhos-Curso/2021-05-DM/time-series.ipynb>

Anexaremos também o arquivo à entrega do trabalho.

# Introdução

Conseguimos uma parceria para obtenção das vendas de uma empresa varejista que tem como principal produto de venda calçados, embora outros itens também sejam vendidos nas lojas.

Os dados fornecidos são de janeiro de 2019 até meado de abril de 2021, sendo um dataset analítico, ou seja, nenhum agrupamento foi realizado na geração do arquivo de exportação (formato csv). O arquivo exportado tem aproximadamente 3,3 GB.

O objetivo do trabalho foi desenvolver modelos de *machine learning* que previssem as vendas para um determinado subconjunto dos dados e janela de tempo definida durante o estudo dos dados. A partir dessas definições os dados seriam preparados e os modelos afinados através do estudo dos melhores hiperparâmetros identificados.

Os modelos tratados em nosso projeto foram *Random Forest* e *SVM*.

# Tratamento dos Dados

O dataset encaminhado tem 3,3 GB e 19.559.118 registros. Os dados são analíticos, cada produto de cada venda realizada desde janeiro de 2019.

Mostraremos nesse tópico o tratamento que realizamos aos dados

## Leiaute Dataset Original

Mostraremos aqui o layout do dataset original que foi encaminhado pela empresa:

* NumeroBoleta – Boleta de Venda, poderia ser utilizado para agrupar itens de uma venda.
* Loja – Loja que realizou a venda. Existe um dataset com o detalhamento de cada loja, mas não foi utilizado em nosso estudo.
* UF – Estado onde o produto foi vendido.
* CIDADE – Cidade onde o produto foi vendido.
* BAIRRO – Bairro onde o produto foi vendido.
* CriacaoReferencia – Data da venda.
* Produto\_Codigo – Código do produto vendido. Existe um dataset de produto, no entanto ele não foi utilizado em nosso estudo.
* SECAO – Primeiro nível de agrupamento dos produtos.
* GRUPO – Segundo nível de agrupamento dos produtos.
* CATEGORIA – Terceiro nível de agrupamento dos produtos.
* COR – Cor do produto vendido.
* Tamanho – Tamanho do calçado vendido.
* Quantidade – Quantidade vendida.
* PrecoVenda – Preço de venda do produto.
* PrecoTranscao – Preço aplicado a venda, quando são diferentes indica uma venda promocional (anersariante do mês, promoção em loja, etc)

## Manipulação dos Dados para Modelos de Machine Learning

Não existiam *missing values* no *dataset*, todos os registros possuíam valores para os campos que importavam para a geração da série histórica.

O *dataset* foi filtrado da seguinte forma:

* Produtos que estejam na seção de Calçados e
* Somente produtos Femininos e
* Vendidos na cidade do Rio de Janeiro.

df\_venda\_preparado = df\_venda[(df\_venda.SECAO == 'CALCADOS') &

(df\_venda.GRUPO == 'FEMININO') &

(df\_venda.CIDADE == 'RIO DE JANEIRO')]

Nosso grupo estudou duas opções de agrupamento dos dados por período:

* Mensal – Vendas agrupadas por mês com forte sazonalidade no período de 11 meses (natal).

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

Gráfico

Descrição gerada automaticamente

* Semanal – Vendas agrupadas por semanas com correlações fortes para 1,2, 5, 6 e 11 semanas anteriores (gráfico de correlação na seção de estatística).

Gráfico

Descrição gerada automaticamente

Gráfico

Descrição gerada automaticamente

Escolhemos a janela semanal que tinha correlações interessantes e aumentaria a quantidade de dados para treinamento do modelo, se optássemos pelo período mensal teríamos poucos dados para treino e validação.

O projeto tem funções para diferentes tamanhos de janelas semanais (12, 24, 36 e 52), dessa forma poderíamos comparar os resultados e encontrar a quantidade de semanas que gerasse a melhor previsão para nossa série.

def monta\_dataset\_por\_janela(dataset=None, window=12):

dataSize = len(dataset)

X = []

y = []

for i in range(window, dataSize):

X.append(dataset.iloc[i-window:i, 1])

y.append(dataset.iloc[i, 1])

X, y = np.array(X), np.array(y)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

return x\_train, x\_test, y\_train, y\_test

Os dados também foram manipulados para tentarmos uma melhoria em relação as janelas das últimas semanas, criamos um *dataset* que continha os seguintes dados:

* 1 e 2 semanas anteriores
* Mesma semana do ano anterior.
* 1 e 2 semanas anteriores do ano anterior.

def monta\_dataset\_ano\_anterior(dataset):

df\_venda\_ml = dataset[dataset.AnoMesSemana >= '2020-01-02'].sort\_values(['AnoMesSemana'])

df\_venda\_agrupado\_ml = df\_venda\_ml.groupby('AnoMesSemana', as\_index=False).agg({

'Quantidade':'sum',

'AnoMesSemana1SemanaAnterior': 'max',

'AnoMesSemana2SemanaAnterior': 'max',

'AnoMesSemanaAnoAnterior': 'max',

'AnoMesSemanaAnoAnterior1Semana':'max',

'AnoMesSemanaAnoAnterior2Semana':'max'})

df\_venda\_ml\_final = pd.merge(df\_venda\_agrupado\_ml, df\_venda\_historico\_agrupado, left\_on="AnoMesSemanaAnoAnterior", right\_on="AnoMesSemana", suffixes=("","\_AnoAnterior"))

df\_venda\_ml\_final = pd.merge(df\_venda\_ml\_final, df\_venda\_historico\_agrupado, left\_on="AnoMesSemanaAnoAnterior1Semana", right\_on="AnoMesSemana", suffixes=("","\_AnoAnterior1Semana"))

df\_venda\_ml\_final = pd.merge(df\_venda\_ml\_final, df\_venda\_historico\_agrupado, left\_on="AnoMesSemanaAnoAnterior2Semana", right\_on="AnoMesSemana", suffixes=("","\_AnoAnterior2Semana"))

df\_venda\_ml\_final = pd.merge(df\_venda\_ml\_final, df\_venda\_historico\_agrupado, left\_on="AnoMesSemana1SemanaAnterior", right\_on="AnoMesSemana", suffixes=("","\_1SemanaAnterior"))

df\_venda\_ml\_final = pd.merge(df\_venda\_ml\_final, df\_venda\_historico\_agrupado, left\_on="AnoMesSemana2SemanaAnterior", right\_on="AnoMesSemana", suffixes=("","\_2SemanaAnterior"))

df\_venda\_ml\_final = df\_venda\_ml\_final[['AnoMesSemana','Quantidade\_1SemanaAnterior','Quantidade\_2SemanaAnterior','Quantidade\_AnoAnterior','Quantidade\_AnoAnterior1Semana','Quantidade\_AnoAnterior2Semana','Quantidade']]

np\_dataset = df\_venda\_ml\_final.to\_numpy()

X = np\_dataset[:,1:-1]

y = np\_dataset[:,-1]

np.random.seed(0)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

return x\_train, x\_test, y\_train, y\_test

Nossos testes e treinamentos foram realizados com as seguintes massas de dados:

* Últimas 12 semanas.
* Últimas 24 semanas
* Últimas 36 semanas
* Últimas 52 semanas
* Mesma semana do ano anterior e 2 últimas semanas desse ano e do ano anterior.

## Análise Estatística

Realizamos análises estatísticas para encontrarmos correlações entre os períodos de janelas e analisarmos os gráficos de previsão de alguns modelos estatísticos conhecidos, como média móvel, amortecimento exponencial e Arima.

No arquivo *jupyter* podem ser encontrados os estudos realizados para o agrupamento mensal, no relatório colocaremos os gráficos semanais, já que foi o período de agrupamento escolhido pelo grupo.

### Média Móvel

Gráfico, Histograma

Descrição gerada automaticamente

Interface gráfica do usuário, Gráfico, Histograma

Descrição gerada automaticamente

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

### Amortecimento Exponencial

Gráfico, Histograma

Descrição gerada automaticamente

### Arima

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

# Teste com Modelos de Machine Learning

Após a preparação dos dados treinamos e validamos os resultados com *Random Forest e SVM*, buscando os melhores hiperparâmetros com *GridSearch.*

def train\_predict\_with\_model(x\_train, x\_test, y\_train, y\_test, model):

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_test)

calcula\_metricas\_erro(y\_pred=y\_pred, y\_test=y\_test,number\_features=x\_test.shape[1])

mostra\_grafico\_previsao(y\_pred=y\_pred, y\_test=y\_test)

# imprimimos os parametros usados no modelo, caso tenha sido usado o GridSearchCV

# para determinar um

if hasattr(model, 'best\_params\_') and model.best\_params\_ is not None:

print(model.best\_params\_)

## Random Forest

**def create\_tuned\_random\_forest\_regressor():**

# veja os parâmetros em:

# https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

tuned\_parameters = {

'n\_estimators': [1, 25, 50, 75, 100, 125],

'criterion': ['mse', 'mae'],

'max\_depth': [None, 1, 2, 3, 4, 5, 6],

'min\_samples\_leaf': [x for x in range(1,11,1)], # de 1 a 10

'random\_state': [0], # para facilitar a reprodução dos resultados observados

'n\_jobs': [-1] # usamos todos os processadores lógicos disponíveis no treino e predição do modelo

}

# com o RandomForestRegressor, não podemos usar o scoring do GridSearchCV

return GridSearchCV(RandomForestRegressor(), tuned\_parameters, verbose=1)

**def train\_predict\_random\_forest(x\_train, x\_test, y\_train, y\_test, best\_model=True):**

if best\_model:

regressor = create\_tuned\_random\_forest\_regressor()

else:

regressor = RandomForestRegressor(n\_estimators = 10, random\_state=0)

train\_predict\_with\_model(x\_train, x\_test, y\_train, y\_test, regressor)

### Dados Ano Anterior

x\_train, x\_test, y\_train, y\_test = monta\_dataset\_ano\_anterior(dataset=df\_venda\_preparado)

%time train\_predict\_random\_forest(x\_train, x\_test, y\_train, y\_test)

*Fitting 5 folds for each of 840 candidates, totalling 4200 fits*

*RMSE: 6380.703473403869*

*MSE: 40713376.8155082*

*MAPE: 81.67389820490126 %*

*R2 Score: -711.3624254417854*

*R2 Ajustado: -1304.9977799766066*

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

### Janela 12 Semanas

x\_train, x\_test, y\_train, y\_test = monta\_dataset\_por\_janela(dataset=df\_venda\_historico\_agrupado, window=12)

%time train\_predict\_random\_forest(x\_train, x\_test, y\_train, y\_test)

*Fitting 5 folds for each of 840 candidates, totalling 4200 fits*

*RMSE: 8139.541838583717*

*MSE: 66252141.34205481*

*MAPE: 175.14516311117572 %*

*R2 Score: -0.8464063713665573*

*R2 Ajustado: -2.8606678674028014*

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

### Janela 24 Semanas

x\_train, x\_test, y\_train, y\_test = monta\_dataset\_por\_janela(dataset=df\_venda\_historico\_agrupado, window=24)

%time train\_predict\_random\_forest(x\_train, x\_test, y\_train, y\_test)

*Fitting 5 folds for each of 840 candidates, totalling 4200 fits*

*RMSE: 14495.618962838393*

*MSE: 210122969.1158*

*MAPE: 121.12492894740994 %*

*R2 Score: -15.718779900480143*

*R2 Ajustado: 118.03145930336102*

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

### Janela 36 Semanas

x\_train, x\_test, y\_train, y\_test = monta\_dataset\_por\_janela(dataset=df\_venda\_historico\_agrupado, window=36)

%time train\_predict\_random\_forest(x\_train, x\_test, y\_train, y\_test)

*Fitting 5 folds for each of 840 candidates, totalling 4200 fits*

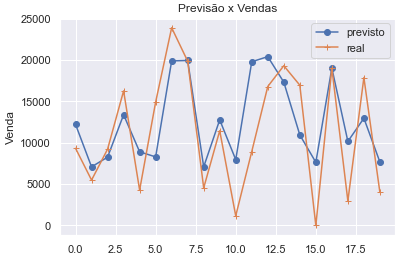
*RMSE: 4871.558085865342*

*MSE: 23732078.18396*

*MAPE: 7702.651779022524 %*

*R2 Score: 0.0020197153103737042*

*R2 Ajustado: 2.1153897299472293*



### Janela 52 Semanas

x\_train, x\_test, y\_train, y\_test = monta\_dataset\_por\_janela(dataset=df\_venda\_historico\_agrupado, window=52)

%time train\_predict\_random\_forest(x\_train, x\_test, y\_train, y\_test)

*Fitting 5 folds for each of 840 candidates, totalling 4200 fits*

*RMSE: 10079.511332568409*

*MSE: 101596548.70337498*

*MAPE: 99.16861134392427 %*

*R2 Score: -12.039859191325553*

*R2 Ajustado: 6.2864294018887374*