

ANÁLISE PREDITIVA COM  
**AZURE  
MACHINE  
LEARNING E R**



DIEGO NOGARE  
THIAGO ZAVASCHI



# ANÁLISE PREDITIVA COM AZURE MACHINE LEARNING E R

*Fazer uma análise preditiva é permitir que suas análises dêem um passo à frente no uso de Business Intelligence convencional.*

*Diego Nogare  
Thiago Zavaschi*



**Dados Internacionais de Catalogação na Publicação (CIP)**  
**(Câmara Brasileira do Livro, SP, Brasil)**

Nogare, Diego

Análise preditiva com Azure Machine Learning e R / Diego Nogare, Thiago Zavaschi. -- São Paulo : B2U Editora, 2016.

Bibliografia.

ISBN 978-85-65904-19-3

1. Computação em nuvem 2. Dados - Análise  
3. Linguagem de programação 4. Tecnologia da informação - Administração 5. Windows Azure  
I. Zavaschi, Thiago. II. Título.

16-07629

CDD-005.74

**Índices para catálogo sistemático:**

1. Azure Machine Learning : Ciência da computação 005.74



*À minha família.*



## Apresentação

Seja bem vindo!

Esta produção tem o objetivo de abrir sua mente para cenários e aplicações que possam dar resultados fantásticos quando aplicados de forma correta. Os autores, Diego Nogare e Thiago Zavaschi, possuem experiência academica e profissional nestas áreas e compartilharão contigo nos 10 capítulos desta obra, os caminhos das pedras para começar e terminar seus projetos utilizando Azure Machine Learning.

Este livro possui uma mistura de teoria e prática, permitindo uma leitura agradável do início ao fim. Ao ler a produção e realizar os exercícios práticos, você terá a chance de fixar o conteúdo e aprender na prática o que um cientista de dados faz.

Independentemente de você ser um profissional que desenvolve software ou que mantém e sustenta um ambiente de infraestrutura, ou um gestor de pessoas e até mesmo um médico, você poderá ter resultados favoráveis para suas atividades se entender os processos e aplicar o aprendizado de máquinas a uma necessidade que enfrenta.

Relaxe, abra seus horizontes e divirta-se nestas próximas quase 350 páginas que foram escritas com muita dedicação e carinho.

## Prefácio

A quantidade de dados gerados hoje pela grande variedade de sistemas e dispositivos é assustadora... empresas, pessoas, indústria, trânsito, medicina, bancos, etc etc etc... todos os segmentos hoje possuem uma capacidade de informações que claramente no passado não se imaginaria. E o mais incrível é que estamos ainda engatinhando, ou seja, em poucos anos isso crescerá exponencialmente.

Onde está o desafio? As oportunidades?

O dado está ai! Dificilmente você não tem o dado que você precisa. E se você não tem, provavelmente consegue capturar e em algum tempo já ter uma amostra significativa... e ai vem a oportunidade: e agora, o que faço com isso?

Essa é **A pergunta!**

Hoje conseguir essa resposta, tirar alguma informação que vale ouro, pode ser sim um grande diferencial competitivo.

Falando um pouco da vida real, sou responsável pelo time de Tecnologia do Grupo Minha Vida, onde atingimos todos os meses mais de 20 milhões de pessoas com diversas plataformas digitais, uma delas é o Dieta e Saúde. Um app onde os usuários adicionam todas as refeições com mais de 300 milhões de refeições inseridas por ano. Esse dado nos ajuda a entender melhor o comportamento da dieta das pessoas, oferecer sugestões de cardápios muito mais personalizados, escalar uma metodologia a partir do conhecimento real do consumo das pessoas... é um cruzamento de dados insano que nenhum profissional conseguiria manualmente capturar e muito menos analisar. Isso é só um exemplo de muitos que já praticamos hoje e de verdade mudamos de ponta cabeça nossos produtos e negócios... o fato curioso é que essa visão é nova e me lembro de anos atrás cogitamos "expurgar" dados do passado... "esse dado não serve para nada mesmo"... ainda bem que não fizemos isso :)

As tecnologias estão ai! Os dados também! Agora é a sua vez.

Neste livro você terá a oportunidade de aprender as principais tecnologias do mercado para dar os primeiros passos nessa exploração! Você aprenderá através de um conteúdo de simples leitura e com exemplos práticos um conteúdo ainda raro.

Sem dúvida alguma você terá um diferencial no mercado enorme, uma vez que ainda é um mercado com poucos profissionais e com uma demanda do mercado gigantesca.

Recomendo a leitura deste livro para todo profissional de Tecnologia, mesmo os que não vão de fato trabalhar no dia-a-dia como um cientista de dados ou analista de BI ou algo do gênero, pois é um conhecimento que sem dúvida alguma será útil para todas as áreas.

**Alexandre Tarifa** – Diretor de TI do Grupo Minha Vida

# Sumário

<b>1 – O DIA-A-DIA DO CIENTISTA DE DADOS.....</b>	<b>17</b>
Por onde começar?.....	18
Evolução contínua para solidificar o conhecimento .....	23
Mas, afinal, o que é um Cientista de Dados? .....	37
<b>2 – INTRODUÇÃO À APRENDIZAGEM DE MÁQUINA E AO AZURE MACHINE LEARNING .....</b>	<b>44</b>
O que é Aprendizagem de Máquina?.....	44
As Motivações para Aprender .....	45
Conceitos Fundamentais.....	46
Características .....	46
Classes.....	47
Aprendizagem Supervisionada e Não-Supervisionada .....	48
Azure Machine Learning – AzureML.....	50
Conhecendo o AzureML Studio.....	52
Aba Projects .....	53
Aba Experiments .....	53
Aba Web Services .....	54
Aba Notebooks.....	54
Aba Datasets .....	55
Aba Trained Models.....	57
Aba Settings .....	58
Botão + NEW .....	59

<b>Módulos do Azure Machine Learning .....</b>	<b>60</b>
<b>3 – TRABALHANDO COM DADOS EXTERNOS .....</b>	<b>63</b>
<b>A importância dos dados .....</b>	<b>63</b>
<b>Obtenção de dados.....</b>	<b>73</b>
<b>Lendo dados externos ao Azure Machine Learning.....</b>	<b>75</b>
<b>Escrevendo dados a partir do Azure Machine Learning.....</b>	<b>82</b>
<b>4 – CRIANDO O PRIMEIRO MODELO .....</b>	<b>90</b>
<b>Identificando o problema .....</b>	<b>90</b>
<b>Componentes necessários .....</b>	<b>91</b>
<b>Criando o primeiro modelo.....</b>	<b>93</b>
<b>Organizando os Experimentos em Projetos .....</b>	<b>119</b>
<b>5 – VALIDANDO UM MODELO NO AZUREML.....</b>	<b>123</b>
<b>Evaluate Model .....</b>	<b>123</b>
<b>Validando Modelos.....</b>	<b>125</b>
Matriz de Confusão.....	125
Acurácia .....	128
Precisão.....	129
Recall .....	129
F1-Score .....	130
Curva ROC e AUC.....	130
Precision/Recall e Lift.....	132
<b>Modelos de Regressão.....</b>	<b>133</b>

<b>Cross Validate Model.....</b>	<b>134</b>
<b>Módulo Partition and Sample.....</b>	<b>138</b>
Sampling .....	139
Assign to Folds.....	140
Pick Fold.....	140
Head .....	141
<b>6 – EXPONDO MODELOS ATRAVÉS DE WEB SERVICES.....</b>	<b>143</b>
<b>Introdução e publicação dos web services do Azure Machine Learning .....</b>	<b>143</b>
<b>Configuração dos web services publicados .....</b>	<b>147</b>
Aba Dashboard .....	147
Aba Configuration.....	150
<b>Alterando os Parâmetros do Web Service.....</b>	<b>152</b>
<b>APIs de Retreino – Retraining APIs .....</b>	<b>156</b>
<b>7 – INTRODUÇÃO À ALGORITMOS DE MACHINE LEARNING .....</b>	<b>160</b>
<b>Introdução aos grupos de algoritmos .....</b>	<b>160</b>
<b>Modelos de Regressão.....</b>	<b>163</b>
Regressão linear .....	163
Regressão Logística.....	169
<b>Classificação .....</b>	<b>172</b>
<b>SVM - Support Vector Machines .....</b>	<b>173</b>
Support Vector Machines para problemas não binários.....	178
Árvore de Decisão, Decision Trees, e Árvores de Decisão Otimizadas, Boosted Decision Trees .....	180
Redes Neurais.....	189
<b>Cluster K-Means .....</b>	<b>199</b>
AGRUPAMENTO DOS DADOS .....	199

ENTENDENDO COMO FUNCIONA O ALGORITMO .....	204
<b>ESCOLHENDO A QUANTIDADE DE K (CLUSTERS) NO ALGORITMO.....</b>	<b>209</b>
<b>Detecção de Anomalias .....</b>	<b>215</b>
Support Vector Machine de Uma Classe.....	216
Detecção de anomalia baseado em Principal Component Analysis .....	220
<b>Caminho das pedras na escolha do algoritmo.....</b>	<b>224</b>
<b>8 – INTERAGINDO COM MODELOS ATRAVÉS DE NOTEBOOKS.....</b>	<b>226</b>
<b>Criando um notebook no AzureML.....</b>	<b>226</b>
<b>Integração dos Notebooks ao AzureML .....</b>	<b>235</b>
<b>Interagindo com um dataset do AzureML.....</b>	<b>238</b>
<b>Publicando Web Services.....</b>	<b>241</b>
Consumindo um Web Service .....	244
<b>9 – AUMENTANDO AS POSSIBILIDADES COM R .....</b>	<b>247</b>
<b>IDE para desenvolvimento.....</b>	<b>247</b>
<b>Utilizando a Linguagem R – Conceitos e Estruturas.....</b>	<b>248</b>
Expressões Aritméticas e Strings .....	249
Variáveis.....	250
Funções.....	251
Executando Arquivos .....	254
Vetores.....	255
Matrizes .....	259
Plotando Vetores e Matrizes .....	261
Data Frames .....	269
<b>Instalando Novos Pacotes.....</b>	<b>277</b>
Instalando e Usando um Pacote do CRAN .....	277

<b>Exemplos utilizados no Capítulo 1 – Dia-a-Dia do Cientista de Dados .....</b>	<b>281</b>
Regressão Linear.....	281
Histograma e Diagrama de Caixa.....	286
Clusterização .....	290
<b>Azure Machine Learning com R.....</b>	<b>300</b>
O Módulo “Execute R Script” .....	300
Utilizar um script R codificado diretamente no componente .....	301
Utilização de um script codificado fora da plataforma .....	303
Utilizando uma biblioteca externa.....	307
Interagindo com datasets externos .....	308
Criando um modelo de aprendizagem de máquina dentro do módulo Execute R Script.....	310
Visualizando todas as bibliotecas do R instaladas no AzureML.....	311
O módulo Create R Model.....	311
Configurando o módulo Create R Model para criar um modelo de Naïve Bayes .....	313
<b>10 – VISUALIZAÇÃO DE DADOS .....</b>	<b>316</b>
<b>CONCLUSÃO .....</b>	<b>346</b>

## 1 – O dia-a-dia do cientista de dados

O papel do cientista de dados na sociedade, nos dias em que esta publicação foi escrita, ainda recebe uma aura endeusada por ser algo considerado novo e complexo. Porém, essa roupagem de utilizar o termo “cientista” com toda sua pompa e glamour só fez com que aumentasse esse misticismo ao redor da profissão. Imagine o cenário de uma análise de genes do DNA humano utilizando redes neurais com alguns bilhões de observações (conhecidos como linhas [ou registros] dentro da área de banco de dados) contendo algumas centenas de variáveis (conhecidas como colunas no banco de dados); isso parece ser enorme e deve dar um trabalho hercúleo para resolver. Alguém para trabalhar com esse cenário precisa ser um CIENTISTA. Ledo engano! Estatísticos enfrentam cenários similares em diversas ocasiões durante suas carreiras e isso não os torna semideuses com toda essa aura envolvida por não terem o título de cientista em sua carteira de trabalho. O papel do cientista de dados é, sim, analisar, destrinchar, esmiuçar, cruzar e porque não também caçar respostas nessa massa de dados. Estatísticos e matemáticos já fazem isso há anos, e por uma atuação pesada das áreas de marketing no mercado, hoje existem vários cientistas de dados de profissão que não são estatísticos ou matemáticos por formação, mas que exercem muito bem este papel no seu dia-a-dia. O objetivo deste capítulo é apresentar diversas técnicas que serão detalhadas em capítulos futuros, processos e ações que devem ser trabalhados pelo profissional que está usando modelos preditivos estatísticos para exercer sua função dentro da necessidade do seu negócio.

Não desista porque você não é formado em física, por não ter um entendimento grande da matemática ou em estatísticas para saber a diferença entre média e mediana. Todos nós começamos nossos estudos em algum momento da vida e, neste momento, estamos todos praticamente no mesmo nível. No final deste livro espero que tenha aprendido todas as técnicas apresentadas e, então, terá insumos para exercer melhor suas atividades diárias.

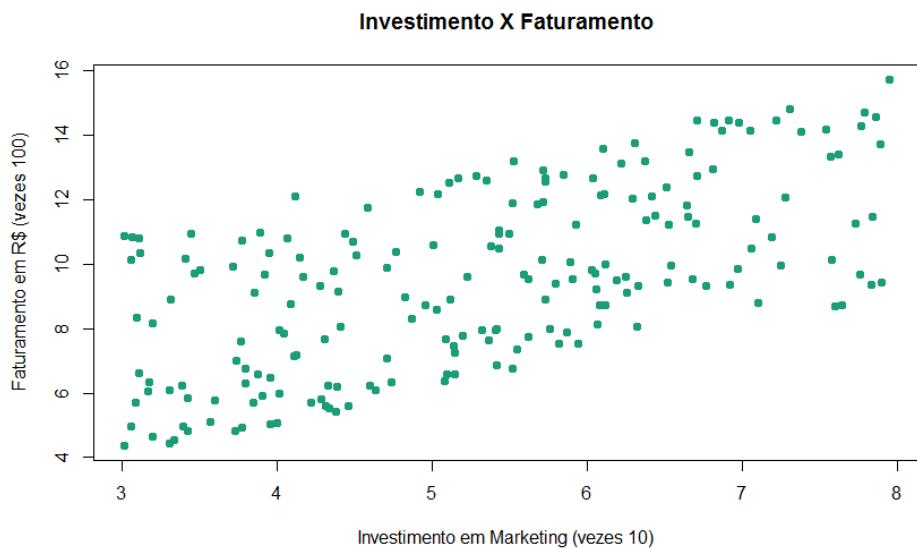
## Por onde começar?

Considero um bom ponto de partida sempre olhar os dados. Não é necessário olhar todo o universo de dados a ser analisado, mas apenas uma parte dele para que você tenha uma ideia de como estão estes dados e de como pode dar o próximo passo. Com essa visão inicial para conhecer a amostra dos dados seu papel será definir quais caminhos serão seguidos para poder responder as perguntas que lhe fizeram. Muitas vezes, as perguntas parecem vagas e superficiais, mas isso não tira sua seriedade. Seus clientes, internos ou externos pedirão respostas para perguntas do tipo "Como faço para vender mais?" Ou "Qual o melhor produto para mim?" Ou então "Quais são os perfis que mais utilizam este serviço?". Todas estas respostas podem ser dadas através da análise de dados, e isso é uma das suas atribuições.

Para responder à pergunta "Como posso vender mais?", devem ser analisados dados de vendas do passado para encontrar o padrão de comportamento que fez as vendas aumentarem. Este padrão pode representar que o aumento em campanhas de marketing no fator de 5% ao mês nos últimos 18 meses fez as pessoas comprarem mais produtos. Ou então o padrão encontrado pode apresentar que as vendas aumentaram por causa de novos clientes que começaram a frequentar a loja e consumiram mais. A resposta encontrada para uma massa de dados A não é, necessariamente, a mesma resposta se analisada a massa de dados B, por mais que sejam da mesma rede de lojas e estejam situadas na mesma cidade. Depois de descobrir o motivo do aumento de vendas do passado, estima-se que continuar realizando as mesmas ações de sucesso faz com que o crescimento continue seguindo a mesma ordem de grandeza, correto? Não necessariamente! Existem fatores externos a essas análises iniciais que os dados apresentaram. Quais fatores externos podem impactar os números que vinham crescendo linearmente nos últimos meses e os fizeram desacelerar mesmo seguindo os mesmos investimentos em marketing? Será que o preço do barril de petróleo impacta minhas vendas, ou então será as chuvas que alagaram os campos de produção de feijão fizeram o preço do meu produto disparar e isso diminuiu as vendas?

Descobrir estes agentes também é o seu papel. Pensar, pensar e pensar. Quais ações ou produtos estão impactando o crescimento fará o CEO, que quer saber como vender mais, confiar ou não nas suas análises futuras. Até este momento, no cenário de aumento de vendas, o que foi feito foi um forecast padrão, pois segue somente os dados do passado para avançar a projeção do futuro com o mesmo fator de crescimento. Depois de definir as variáveis que fizeram parte desta descoberta de dados, e qual técnica pode ser aplicada neste modelo, será possível responder uma pergunta que avança a partir da anterior. Imagine a cena da apresentação do resultado para seu CEO, e então ele te fala: "Parabéns! Obrigado por mostrar qual é o fator que nos faz aumentar a venda. Agora que sabemos que o valor de investimento em marketing faz nosso faturamento aumentar. Quero saber: Qual será o faturamento se eu investir R\$80 em marketing?". Como um cientista de dados pode responder a esta pergunta?

Uma das técnicas que pode ser aplicada para responder a essa pergunta é o uso de regressão linear, na qual um gráfico de dispersão é criado com dados existentes apresentando todas as relações de investimento e faturamento. Neste caso, em que se é preciso responder qual será a venda quando há aumento no investimento em marketing, pode-se dizer que o FATURAMENTO depende do INVESTIMENTO. Colocando estes termos em sintonia com os termos usados em pesquisa é correto dizer que o faturamento será a variável dependente e o investimento será a variável independente porque o faturamento depende do investimento. Quando aplicamos esse modelo em uma regressão linear o eixo X é sempre representado pela variável independente, no caso do exemplo o INVESTIMENTO; e o eixo Y representa a variável dependente, o FATURAMENTO. Um gráfico de dispersão desta correlação entre investimento e faturamento pode ser observado na Figura 1.



*Figura 1 - Amostragem de Investimento X Faturamento*

Para criar a regressão linear, o exemplo coletou todas as amostras de Investimentos Vs. Faturamento do passado e criou a linha que melhor representa a média de pontos entre os valores. Veja na Figura 2 esta linha de regressão.

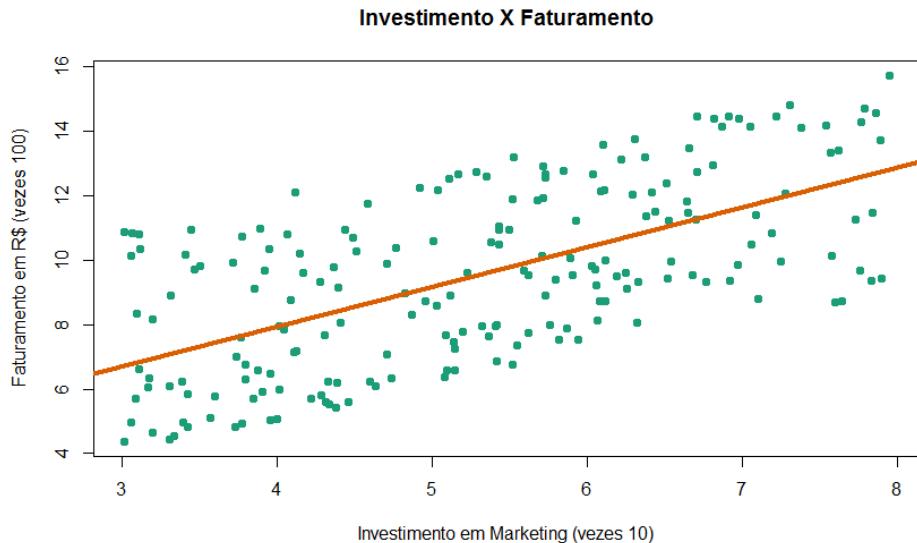


Figura 2 - Linha de regressão do Investimento X Faturamento

Após criar a linha da regressão, agora é possível prever qual será o valor de faturamento baseado no investimento. Para isso, trace uma linha vertical a partir do eixo X até a linha de regressão e em seguida encontre o valor no eixo Y. Para calcular esta correlação é necessário utilizar o que é chamado de Coeficiente de Correlação, no qual uma das possibilidades de cálculo é dada pela multiplicação dos pares, e em seguida é calculada a média dos resultados. Essa multiplicação dos pares é feita com a multiplicação da variável X pela Y da primeira observação, em seguida multiplica-se a variável X pela Y da segunda observação, e assim sucessivamente até terminar toda a amostra. Por fim, é calculada a média destes resultados e com isso obtém o Coeficiente de Correlação. Quando este coeficiente está mais próximo de -1 ou 1 ele tende a ser melhor, quando está mais próximo de 0 ele tende a ser pior. A partir disso é necessário encontrar outras duas variáveis que são importantes para a regressão, a interceptação e a inclinação. Para finalizar esta estimativa de valores preditivos, é necessário utilizar uma equação linear para a regressão que foi proposta. Uma das equações amplamente utilizadas é:

$$y = a + bx$$

Onde  $y$  é a variável dependente;  $a$  é a interceptação da regressão logística;  $b$  é a inclinação da reta, e  $x$  é a variável independente.

Ao aplicar os valores calculados e estimados na equação linear, será possível responder ao seu CEO qual será o faturamento esperado com o investimento planejado. A Figura 3 mostra o ponto de cruzamento do valor 8 do eixo X com a reta da regressão e uma reta do eixo Y até este mesmo ponto na reta de regressão. O valor do eixo Y apresentado é de 7,09. Lendo os títulos dos eixos, o eixo X possui os investimentos em marketing dividido por um fator de 10 e o Y o faturamento dividido por um fator de 100. Ao recalcular estes valores pelos fatores de divisão, será possível dizer ao certo os valores reais, sendo o investimento R\$ 80,00 e o faturamento de aproximadamente R\$ 709,00.

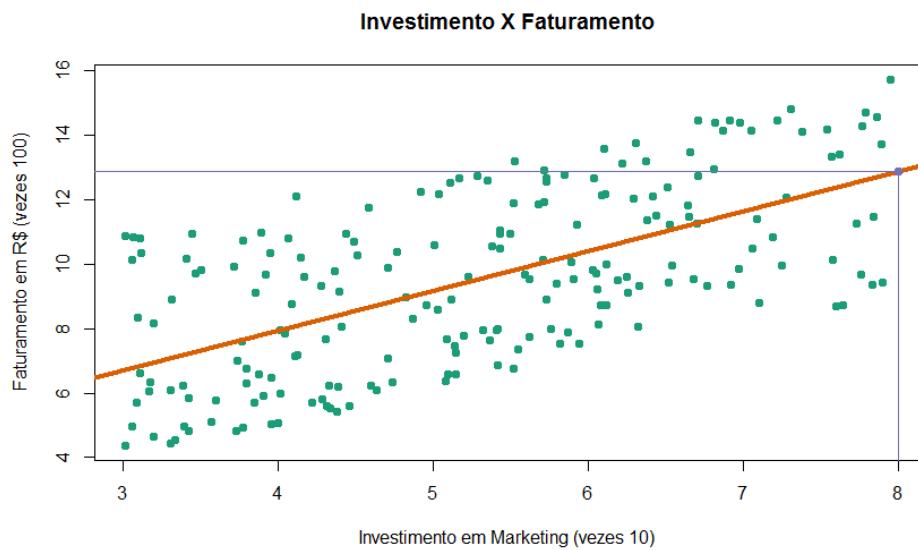


Figura 3 - Valor de Faturamento com base no Investimento

Não se preocupe em tentar entender a fundo a regressão logística neste momento, este processo será explicado com muitos detalhes nos

próximos capítulos, onde serão abordados os principais algoritmos e modelos preditivos para se trabalhar como cientista de dados. Este é só um exemplo da possibilidade de se responder uma pergunta que parece simples, mas que é bastante relevante.

## Evolução contínua para solidificar o conhecimento

É importante e recomendado que os métodos estatísticos estejam claros para um cientista de dados, pois estes métodos são utilizados com muita frequência nas resoluções de problemas do cotidiano. Um exemplo disso é usar histogramas para representar visualmente uma amostragem de dados. Antes de criar o histograma, deve-se entender quais dados serão plotados no gráfico. Existem dois tipos de dados que podem ser usados, dados categóricos e dados numéricos.

Dados categóricos são dados que não podem ser medidos, enquanto dados numéricos podem ser medidos. Para facilitar, mentalize um formulário de avaliação qualquer onde existem apenas duas perguntas. Uma que você deverá colocar a pontuação de 1 a 5 escolhendo a quantidade de estrelas que aquele serviço prestado merece, e o segundo campo é aberto onde você informa sua idade. As estrelas são consideradas dados categóricos e o campo idade é considerado dado numérico. Não é possível medir as respostas da pontuação das estrelas, mas a idade é possível. Para esclarecer essa diferença, dados numéricos (os que podem ser medidos) possuem métricas conhecidas, como idade, peso, altura, etc. Os intervalos daquela medida são conhecidos e inalterados. A idade é medida em anos, e sempre um ano é um ano. Isso será igual independente de quem esteja medindo. Sempre depois de 18 anos de idade, vem a idade de 19 anos. Seja no Brasil ou no Japão. Já os dados categóricos, que não podem ser medidos, dependem da experiência e conhecimento vivido de cada um que responde àquele questionário. Sua avaliação será diferente da minha, por mais que tenhamos o mesmo serviço prestado pelo mesmo fornecedor no mesmo momento. Pense neste formulário de resposta sendo a classificação que você daria para um filme clássico como "E o vento levou" e como um cinéfilo de 65 anos o faria. Agora

avalie um filme como “Star Wars Episódio VII” e convide este mesmo cinéfilo para o avaliar. Repare que a experiência e a vivência de cada um interferem diretamente no resultado.

Voltando ao histograma, ele pode ser criado a partir de dados numéricos sendo uma representação gráfica que permite que dados sejam analisados baseados na média de ocorrências definidos por um intervalo. Imagine que você foi contratado para uma loja virtual que comercializa produtos que variam de R\$1,00 a R\$500,00 reais. É possível separar os produtos em categorias de faixas de preço, sendo a categoria 1 os produtos de R\$1,00 a R\$100,00 reais, a categoria 2 os produtos de R\$101,00 a R\$200,00 reais, e assim até a categoria 5 que varia de R\$401,00 a R\$500,00. Visualizando estes grupos com suas respectivas médias de preço, é possível encontrar uma representação visual como apresentado na Figura 4, onde o eixo X do gráfico representa o crescimento de valores e o eixo Y a frequência de aparições de produtos daquela faixa de preço na amostragem.

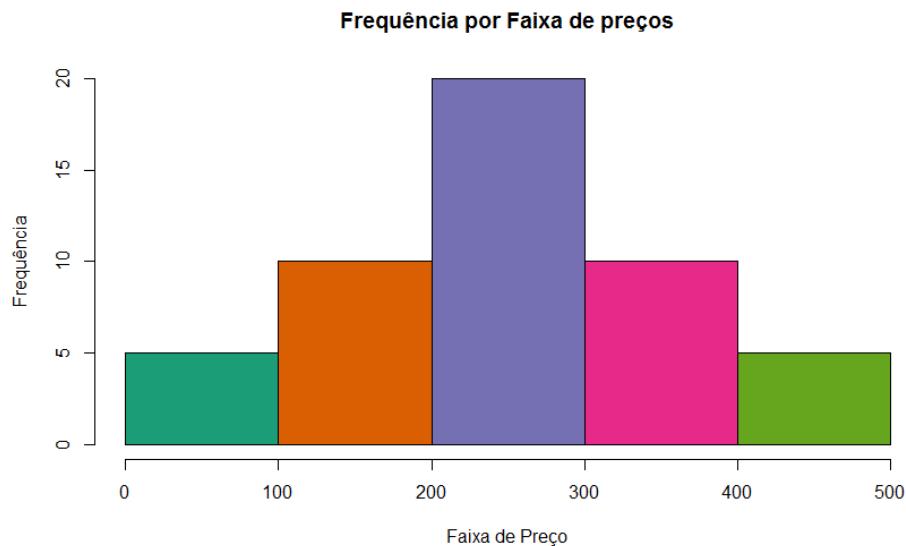


Figura 4 - Histograma de frequência por preços

Com essa visualização, é possível observar que a maioria dos produtos está na faixa de preços entre R\$201,00 e R\$300,00 reais. Certo, mas então qual é a média de valor dos produtos em cada faixa de preço, ou então qual é a mediana, ou o valor mínimo e valor máximo? Todas estas respostas fazem parte de uma área da estatística chamada Médias de Posição, que retornam valores calculados da amostragem que são sempre utilizados por analistas por serem respostas básicas e muito úteis. Cálculos como Média Aritmética, Mediana, Moda, Quartis, Percentis, e alguns outros. Estes cálculos podem ser feitos manualmente entendendo como cada um dele é feito, ou então, utilize-se algoritmos computacionais já escritos por outros profissionais que cedem seus códigos. Independentemente da forma que foi usada para calcular estes métodos, existe uma outra representação gráfica que possibilita uma análise mais assertiva e direta do que somente olhar a massa de dados. Esta representação é conhecida como Diagrama de Caixas, que apresentam os dados de Menor Valor, Primeiro Quartil, Mediana, Terceiro Quartil e Maior Valor. Entenda onde estes valores são impressos no Diagrama de Caixa observando a Figura 5.

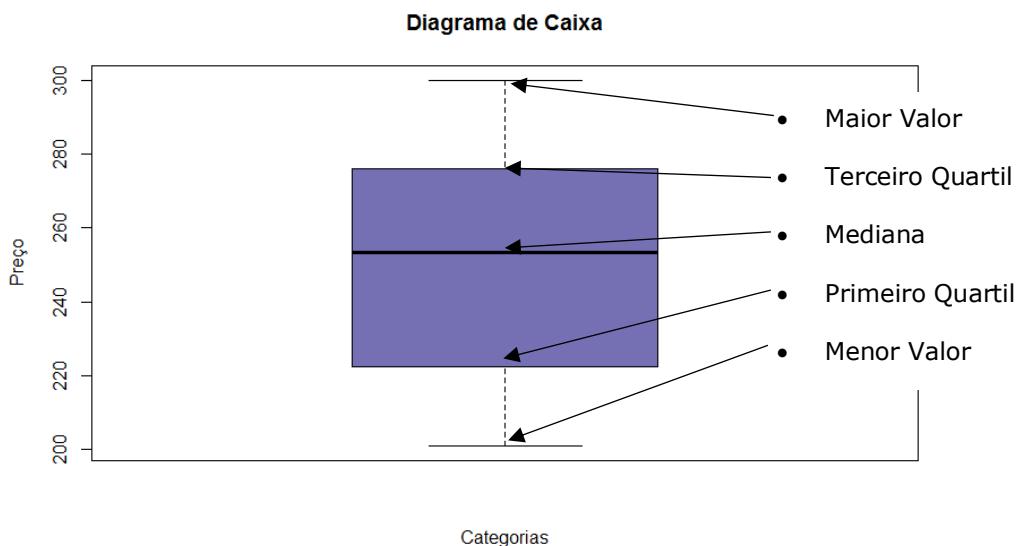


Figura 5 - Valores apresentados por um diagrama de caixa

Depois de entender os dados que são impressos em um diagrama de caixa, vale estender e realizar a análise para toda a amostragem. Isso ajudará a fazer descobertas nos dados de forma a melhorar a tomada de decisão. Acompanhe a Figura 6 como são apresentados os dados de cada faixa de preço no exemplo da loja virtual que foi usada para geração do histograma.

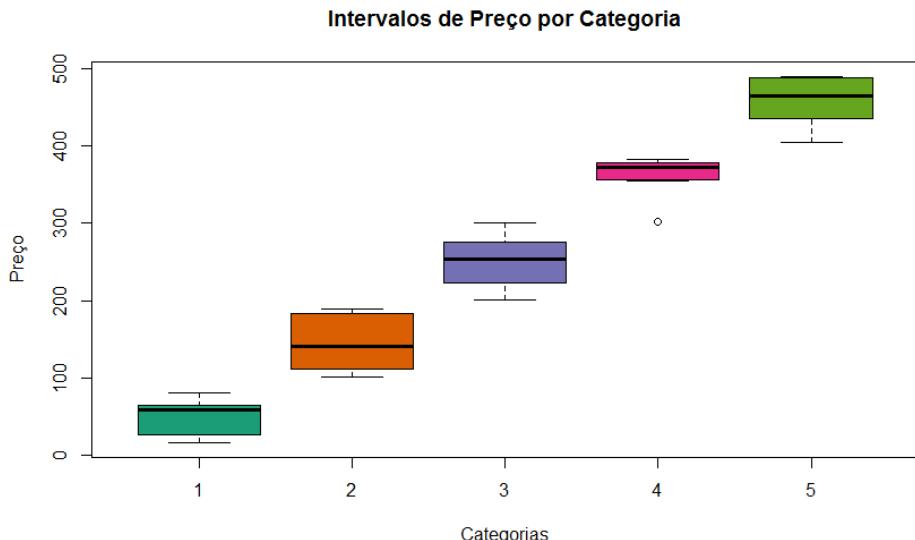


Figura 6 - Diagrama de caixa por faixa de preço

Observando os dados de um gráfico como esse e sabendo como é o cálculo de cada um dos valores estatísticos apresentados, é possível entender que a mediana dos preços da categoria 1 e 4 são muito próximas do terceiro quartil de valores. As categorias 3 e 5 possuem a mediana praticamente no centro entre os primeiro e terceiro quartis e a categoria 4 sendo uma categoria que possui valores relativamente próximos entre si. Repare que na categoria 4 existe um ponto muito abaixo da caixa, praticamente na direção da posição 300 no eixo Y. Este ponto no gráfico é conhecido como *outlier*, que é um dado que diverge do conjunto analisado e, geralmente, direciona o analista a removê-lo da análise para não impactar como um ruído no resultado esperado.

Outros processos amplamente estudados e conhecidos por cientistas de dados são técnicas de aprendizado de máquinas nos três níveis, a saber: Supervisionados, Não Supervisionados e Semi-supervisionados. O aprendizado supervisionado é informado ao algoritmo que fará a busca de padrões o que deve ser encontrado, que então passa a procurar por padrões semelhantes àquele informado e encontra os elementos que se assemelham ao que foi solicitado. O aprendizado semi-supervisionado trabalha com reforços positivos e negativos de acordo com o desempenho do algoritmo. Podendo melhorar a busca pelo padrão quando encontra o resultado positivo recebendo um reforço bom, e quando encontra algo que diz pertencer ao padrão e na verdade não é, ele recebe uma penalidade para aprender que aquele valor não faz parte daquele grupo. Já o aprendizado não supervisionado o cientista de dados não diz explicitamente o que o algoritmo deve procurar, este por sua vez aplica algumas técnicas de agrupamentos para encontrar padrões de semelhança entre os dados e separa na quantidade de grupos definidas. A partir deste momento, qualquer outro elemento que entrar na massa de dados que estão sendo analisadas passa a fazer parte de um dos grupos já existentes.

Acompanhe uma técnica de aprendizado de máquina não supervisionada através do método K-Means (ou K-Média, em português), onde o K representa a quantidade de clusters (conglomerados) que o algoritmo irá receber e agrupar com base nos dados. Para entender este funcionamento continue a imaginar o cenário apresentado no exemplo do histograma da loja virtual. Porém, nesta técnica você não foi apresentado para nenhuma categoria, você só sabe que existem produtos com seus valores e também um índice de vendas, que não será utilizado para nada. Como você pode agrupar estes produtos dentro de 5 categorias?

Ao gerar o gráfico desta amostra é possível ver os elementos conforme a Figura 7

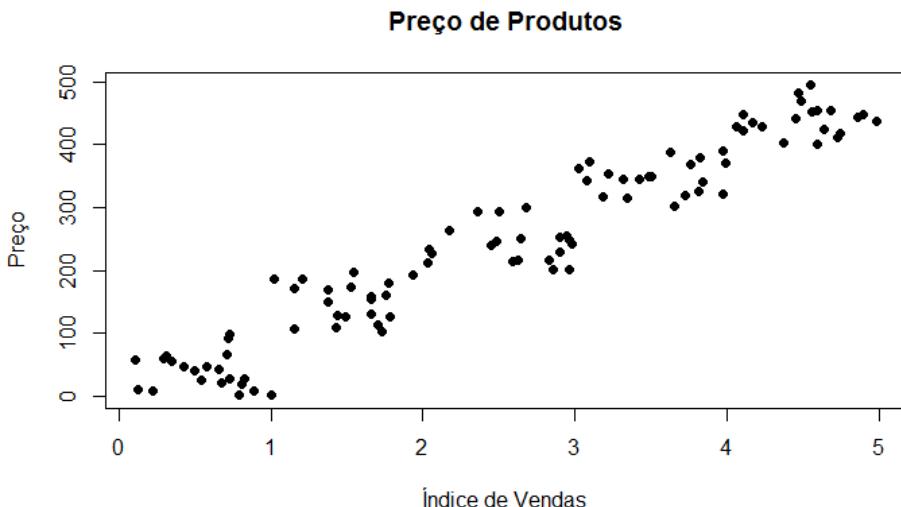
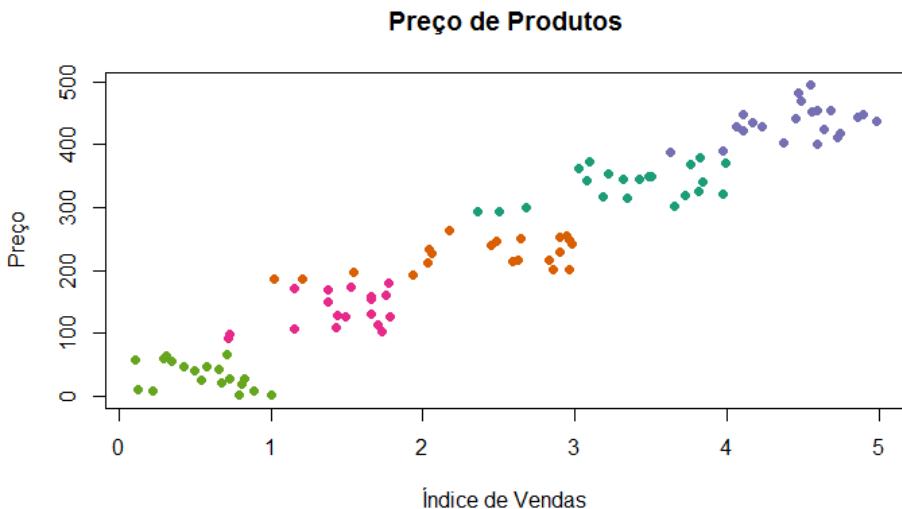


Figura 7 - Preço dos produtos sem categorias

Sabemos que 5 categorias devem agrupar os valores, logo, o valor para K utilizado no algoritmo será de 5. Conhecendo isso, o algoritmo irá colocar aleatoriamente 5 pontos em posições distintas no gráfico. Ao colocar estes pontos aleatórios, o algoritmo calcula a distância média dos pontos existentes para estes 5 centroides que irão representar os grupos. Após este cálculo das médias, os pontos do centroide se movem para a posição que representa a média dos valores dos pontos que estavam mais próximos a ele. Novamente o cálculo das distâncias é feito e é verificado se algum ponto dos dados trocou de centroide por algum que está mais próximo a ele depois da movimentação. Isso fica em um loop até nenhum ponto trocar de grupo depois da movimentação do centroide. Neste momento o algoritmo considera que encontrou o ponto central de cada grupo.

Com estes grupos definidos, é possível imprimir novamente o gráfico e ver como ficaram as separações dos dados, em cada um dos 5 grupos definidos inicialmente. Veja como ficou isso na Figura 8 abaixo.



*Figura 8 - Dados separados em cinco grupos*

Também é possível apresentar ao gráfico as posições de cada centroide após o término do cálculo do algoritmo, isso define o porquê dos grupos terem sido segmentados da forma que estão plotados no gráfico. Acompanhe a Figura 9 onde estão cada um deste pontos.

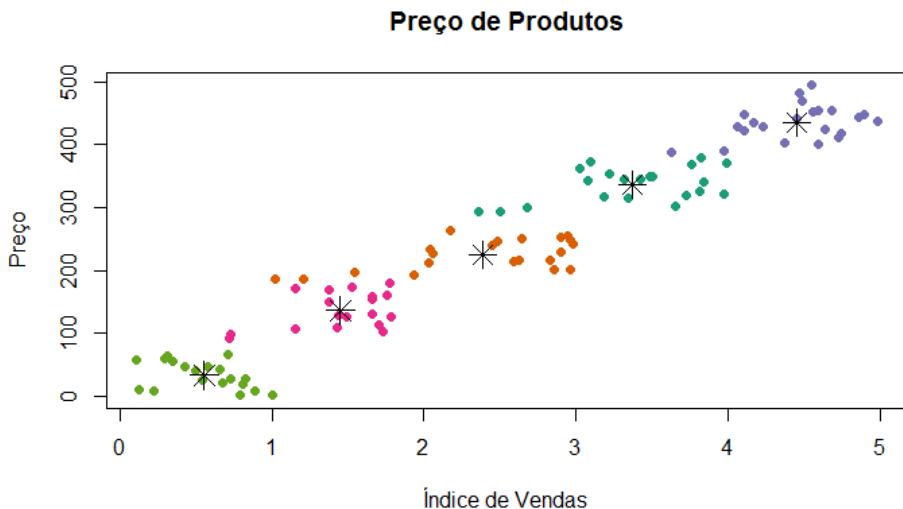


Figura 9 - Posição de cada centroide nos grupos

Com isso foi possível identificar facilmente as 5 categorias de preços que o algoritmo agrupou e, então, você poderá responder ao seu superior quais são as categorias que ele havia lhe pedido informando os elementos que estão em cada grupo.

Também é possível replicar esta técnica de aprendizado não supervisionado para o primeiro exemplo que explicamos neste capítulo, que fala sobre o Investimento X Faturamento utilizando regressão linear. Voltando àquela massa de dados sem separação de segmentos encontra-se algo como apresentado na Figura 10.



Figura 10 - Dados sem separação de Investimento X Faturamento

Agora, ao invés de usar apenas 5 grupos, como no exemplo da loja virtual, sua necessidade é trabalhar com 10 segmentos diferentes para poder responder ao CEO qual campanha de marketing teve melhor performance mesmo trabalhando em uma faixa de investimento semelhante. A separação em dez grupos destes dados apresenta uma segmentação como na Figura 11, veja como ficou.

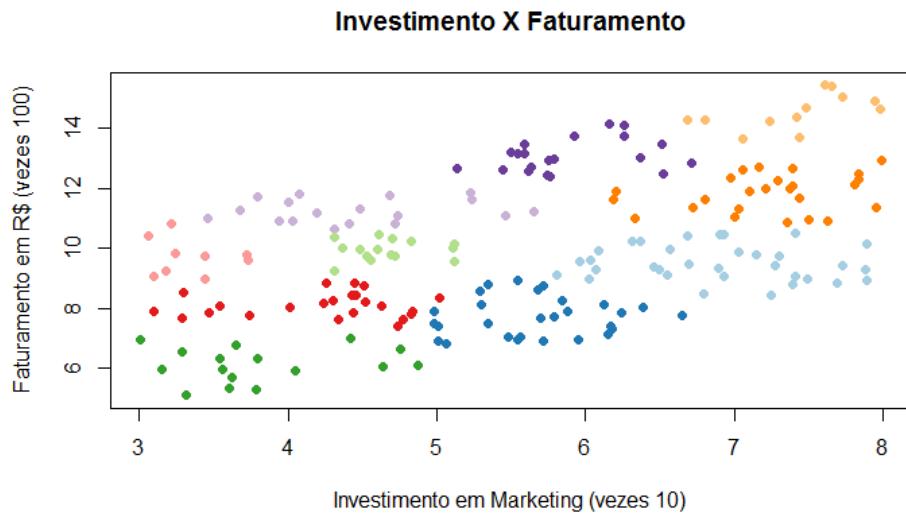


Figura 11 - Separação de Investimento X Faturamento em dez grupos

Para identificar melhor esta segmentação dos grupos também é possível colocar os pontos do centroide neste gráfico e visualizar onde cada ponto de cada grupo encontrou seu centro. Acompanhe os centroides na Figura 12.

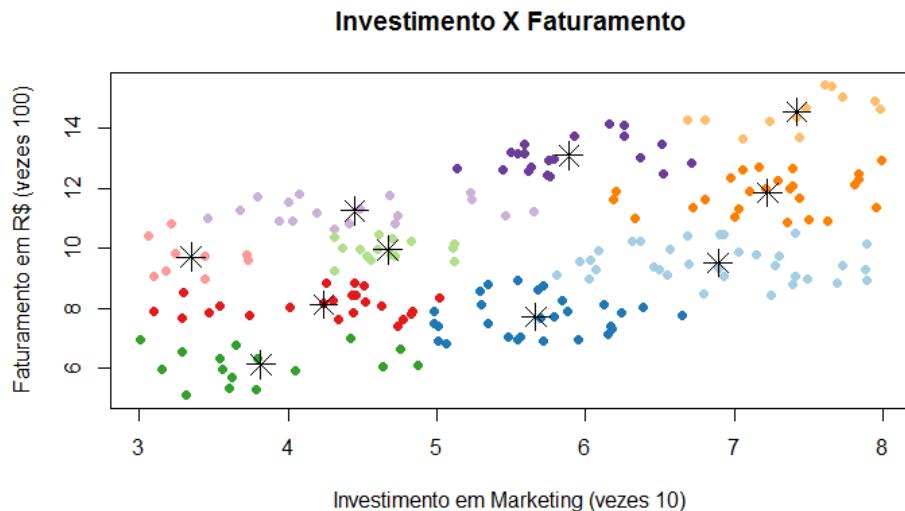


Figura 12 - Centroides de cada um dos dez grupos

Com isso é possível identificar as campanhas de marketing que tiveram mesmo grau de investimento, porém o retorno foi maior. Veja o exemplo do investimento que está no segmento do eixo X entre os valores 5 e 6. É possível ver claramente que existe um grupo em que o faturamento ficou em 7 e 8 no eixo Y (representado pela cor azul) e outro grupo que ficou entre 12 e 14 também no eixo Y (representado pela cor roxa). Isso possibilita ao tomador de decisão poder investir o mesmo valor das campanhas que estão no grupo B, que é entre 50 e 60 reais, porém direcionar para que sejam feitas campanhas similares as que foram utilizadas no grupo A, possuindo um retorno médio 50% maior de faturamento com o mesmo investimento.

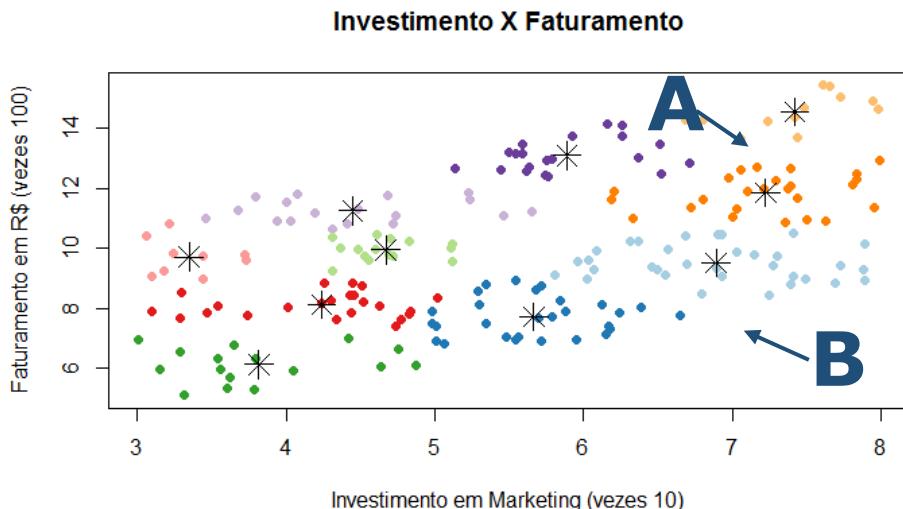


Figura 13 - Grupos A e B semelhantes

E não são só de Regressões ou Clusters que é possível trabalhar com Aprendizado de Máquinas, também existe uma área de atuação conhecida como Classificação, que através de algumas observações é possível dizer em qual classe aquele elemento será posicionado – ou classificado. Imagine um cenário onde é preciso conceder empréstimo financeiro para um solicitante, e com base no comportamento de outros indivíduos que possuem perfis similares ao deste solicitante, é possível classificá-lo entre dois grupos, sendo “Bom Pagador” ou “Mau Pagador”. Com certeza esta classificação é delicada e muitas variáveis de entradadevem ser avaliadas para não cometer injustiça com este solicitante. Posto isso, que é uma classificação que depende de diversas variáveis, criar uma solução através de métodos estatísticos que avaliam os registros históricos de perfil e comportamento de muitos solicitantes de empréstimo é possível inferir que este solicitante de agora é um bom pagador ou não, baseado em sua classificação.

Diversos algoritmos de classificação podem ser testados para realizar esta classificação binária para decidir se o solicitante do empréstimo é um bom ou mau pagador, onde apenas estas duas possibilidades existem.

Também pode-se trabalhar com classificação multi-classes, onde a possibilidade de adequar o valor é maior que duas possibilidades. Voltando ao cenário proposto acima, é possível criar alguns testes e comparar qual algoritmo de classificação binária possui a maior área de cobertura possível entre os algoritmos utilizados. Uma forma de fazer isso é comparando os valores da AUC – Area Under the Curve – Área sob a Curva, que é um valor calculado de qual é o percentual de possibilidades de acerto da predição que se está fazendo. Ao comparar os valores de AUC dos algoritmos testados, quanto maior for a curva mais assertivas serão as previsões. Para chegar na curva AUC é necessário criar o processo de classificação e testar o desempenho dos algoritmos. Acompanhe na Figura 14 um pedaço de um exemplo destes algoritmos dentro do Azure Machine Learning.

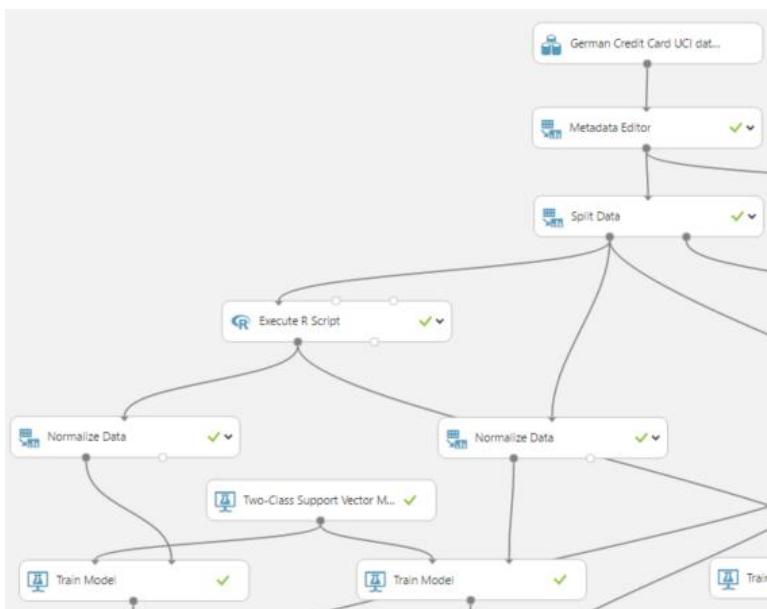


Figura 14 - Parte do fluxo de uma classificação

Após o término do experimento é possível comparar a AUC dos algoritmos testados e encontrar qual tem maior cobertura. Repare na Figura

Figura 15 a curva que cobre a AUC está destacada e representa um AUC de 0.691, isso significa aproximadamente 69% da área do gráfico coberta.

Já na Figura 16 é possível ver a outra curva em destaque, e esta curva cobre aproximadamente 72% do gráfico, sendo eleita a vencedora na comparação com a anterior.

Apesar de ambas curvas cobrirem áreas similares dentro do gráfico é importante utilizar o algoritmo que saiu vencedor na comparação, neste caso, o algoritmo que gerou o gráfico da segunda AUC.

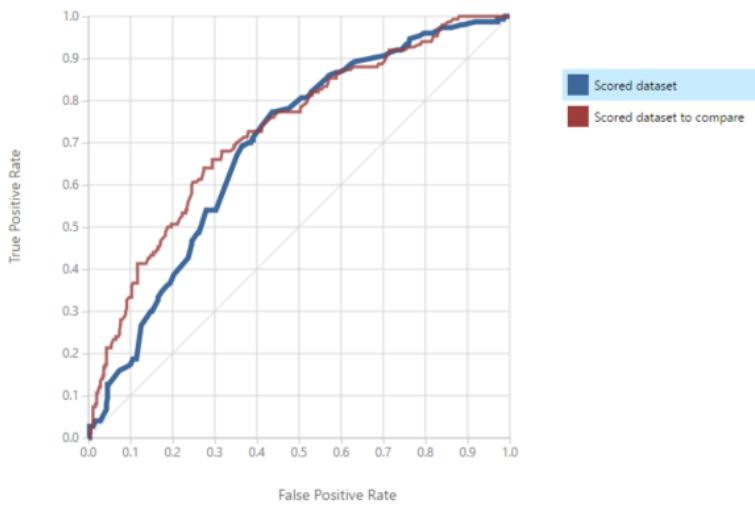


Figura 15 - AUC com cobertura de 69%

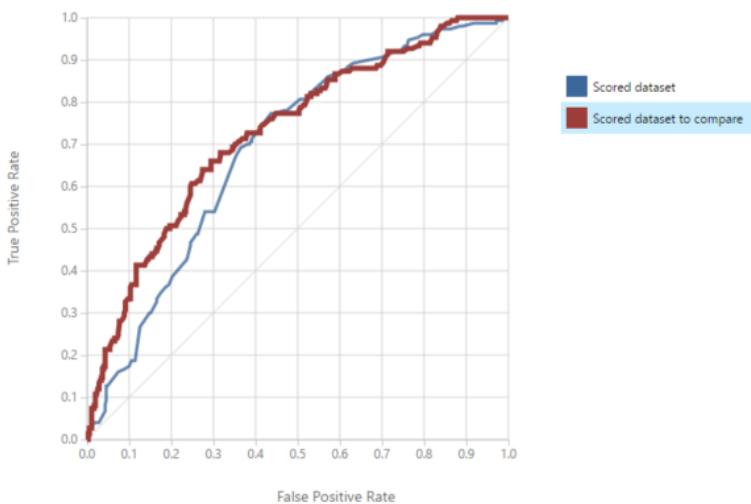


Figura 16 - AUC com cobertura de 72%

Esta comparação de valores de AUC pode ocorrer diversas vezes dentro do processo de escolha do melhor algoritmo para a análise proposta. Após decidir qual algoritmo deve ser utilizado, ele é transformado em um Modelo Preditivo treinado e, então, os dados que precisam ser analisados passam a consultar este modelo que faz o papel de classificar o resultado em “Bom Pagador” ou “Mau Pagador”, comparando os novos dados inseridos com o modelo preditivo treinado.

## Mas, afinal, o que é um Cientista de Dados?

Nas seções anteriores deste capítulo foram apresentados só alguns exemplos de respostas que podem ser obtidas através de técnicas que os cientistas de dados usam. Porém, não é só este tipo de atividade que o cientista faz em cima dos dados e suas análises, também são necessários conhecimentos nas áreas de Matemática, Descoberta de Conhecimento (*KDD* – *Knowledge Discovery in Database*), Visualização de Dados (*Data Visualization*), Aprendizado de Máquinas, Mineração de Dados e algumas

outras. Não só as áreas de atuação são diversas, mas também algumas linguagens de programação como Python, R, S, Java, Ruby, Skala, Octave, Kafka. Avançando um pouco mais neste mundo, chega um momento que sua massa de dados para análise é enorme e os dados não são totalmente estruturados. Neste momento, o uso de técnicas para resolver problemas através de soluções de Big Data passa a ser necessário. Tecnologias como HDInsight, Hadoop, Spark e Storm passam a ser usadas frequentemente e devem estar sempre a postos na sua mala de ferramentas. Ferramentas para expandir o uso de Big Data como Hive, Pig, Mahout, Sqoop, HBase, entre outros tantos começam a te deixar familiarizado com seu uso rotineiro.

A ciência dos dados nos permite trabalhar tanto com um volume de dados pequeno quanto um volume gigantesco, na casa de Petabytes. Sejam eles estruturados ou não, oriundos da nossa empresa ou de fora. O resultado disso é o poder de ter *insights* poderosos que ajudarão a melhorar a tomada de decisão dentro da empresa. Relembre os exemplos anteriores deste capítulo e veja os benefícios!

No fim do dia de trabalho, nosso principal objetivo é transformar dados em conhecimento. Pode ser que sejam usados somente dados estruturados, mas terá situações que os dados virão através de um agente não humano, como um dispositivo que emite sinais sobre telemetria de um carro. Ou sobre o funcionamento do marca-passo que foi implantado em um paciente com problemas no coração. Terá situações em que as análises poderão ser apenas através de ondas de sinais de rádio, ou micro-ondas, ou potência da frequência de um som. Para resolver problemas como estes é necessário separar o sinal do ruído e trabalhar no que é realmente importante descartando o excesso de dados, que são considerados sujos e inúteis. Recentemente, a Microsoft lançou o Skype for Business. Diversas apresentações foram feitas, nas quais dois interlocutores que falavam idiomas diferentes conversavam entre si, em seus respectivos idiomas, e o Skype utilizando técnicas de Processamento de Linguagem Natural fazia a tradução simultânea entre os dois envolvidos. Isso significa que não é mais necessário aprender chinês para fazer uma ligação para China na próxima semana. Pode-se fazer isso através do Skype e continuando a falar português daqui e a pessoa ouve chinês lá na outra ponta, ela responde em chinês e nós ouvimos em português aqui. Isso não é futuro, isso já é presente!

Basicamente, existem quatro grandes áreas para serem analisadas, e isso é amplamente discutido na literatura quando se estuda a ciência dos dados. As quatro áreas são: Análise Descritiva, Análise Diagnóstica, Análise Preditiva e Análise Prescritiva.

**Análise Descritiva:** Ajuda a responder questões sobre o passado. Isso é possível analisando dados históricos de uma determinada situação e geralmente precisa de interação humana para interpretar os dados. Respostas para conhecer o que houve, como por exemplo, para responder “O que aconteceu com os pacientes que tiveram gripe em 2015?” ou então “Quem são os nossos clientes que mais compraram maças em fevereiro?”.

**Análise Diagnóstica:** Ajuda a explicar as situações do passado e o motivo de acontecerem. Ao analisar os dados de um cenário no qual seus clientes estão cancelando a assinatura do seu produto é possível entender o porquê de estarem fazendo isto. Esta análise pode ser feita em conjunto com uma técnica conhecida no marketing como *Churn*, que analisa o engajamento de uma determinada amostra no decorrer do tempo.

**Análise Preditiva:** Ajuda a prever o futuro. Esta análise pode ajudar a predizer com base em um ou mais conjuntos de entradas o que estes dados podem representar. Por exemplo, é possível dizer se uma transação bancária é fraudulenta ou não. Ou então, se um e-mail pode ser considerado SPAM ou um e-mail útil para quem o recebeu.

**Análise Prescritiva:** Ajuda a nortear os próximos passos. Este tipo de processo ajuda o tomador de decisão a escolher um caminho para otimizar seu negócio. Em geral, é combinada com outras análises como, por exemplo, a análise preditiva. Imagine que seu negócio tenha envolvimento com análise de crédito e seus dados dizem que se um solicitador possui certos padrões de entrada (dados pertinentes ao solicitante) e isso representa que este o padrão comportamental do perfil possui uma taxa de inadimplência de mais de 75%.

Esta análise ajudará você a decidir se você fará o empréstimo para este solicitador.

A ciência de dados também pode atuar em uma área que vem recebendo uma exploração nos últimos tempos que é a venda dos dados tratados e não tratados. Algumas empresas enxergaram este nicho como uma forma de faturamento e trabalham com processamento e limpeza de dados. Este nicho ganhou notoriedade nos últimos anos quando grandes fornecedores como Amazon e Microsoft abriram os marketplaces para que empresas pudessem comercializar os dados para o mundo. Com estes marketplaces é possível consumir dados de cotação do dólar, valor do barril de petróleo, entre outros, e com isso, cruzar estes dados externos com os dados da sua empresa e encontrar uma possível causalidade do porque está vendendo mais veículos movidos a Etanol do que a Gasolina. Mas não são só empresas que comercializam dados tratados que viram um diferencial competitivo neste segmento de mercado, as universidades também disponibilizam muitos dados de pesquisa de forma aberta e gratuita, os governos e agências governamentais também o fazem. Veja exemplos dos sites de transparência do governo brasileiro, ou então da NASA que é uma agência do governo americano. Sites como Dados.Gov<sup>1</sup> que possui diversos dados nacionais abertos e gratuitos para uso em seus estudos ou pesquisa, como também o Data.Gov<sup>2</sup> que faz a abertura de dados dos Estados Unidos. Alguns ministérios do Brasil, como o Ministério da Saúde disponibiliza através do DataSus todos os dados de atendimentos do Serviço Único de Saúde de vários anos para que sejam analisados por quem quiser. Isso para nós, cientistas de dados, é um grande universo inexplorado que ajuda nosso trabalho diário.

Com o poder existente na computação em nuvem, o preço baixo e flexibilidade permite que nosso trabalho seja realizado da melhor forma possível. Muitos anos atrás, quando se precisava processar um grande volume de dados os computadores acessíveis ficavam dias, e até semanas para

---

1 <http://www.dados.gov.br>

2 <http://www.data.gov>

processar os dados necessários. Hoje em dia, é possível provisionar um computador com alguns Gigabytes de memória RAM e alguns núcleos de processadores. Isso faz o processamento desta massa de dados diminuir de tempo significativamente e acelera a entrega do resultado. Ao término do processamento, pode-se desligar o computador e não pagar mais pelo uso. Fornecedores como Amazon e Microsoft entregam soluções que atendem estes requisitos sem burocracia e num valor totalmente acessível para quem precisa trabalhar nestes cenários.

Uma forma simples de enxergar o nosso trabalho diário é seguir um roteiro aceito por grande parte dos cientistas. Este roteiro é cíclico e começa com a identificação do problema na área solicitante, passa para o processo de coletar e limpar os dados, em seguida é desenvolvido o modelo preditivo, após o desenvolvimento o modelo é publicado e por fim deve-se monitorar a performance do modelo criado. O processo volta ao passo inicial que é identificar o problema com a área solicitante e entra em um loop sempre melhorando a cada iteração. Um pouco mais de detalhes destes processos estão descritos a seguir. É importante ter esse processo entendido completamente, pois serão explorados em cada experimento dos capítulos a seguir quando os exemplos forem criados.

**Passo 1 – Identificação do Problema:** Este passo é o mais importante do processo como um todo. É a partir daqui que a área solicitante apresenta o problema e nós devemos partir desta solicitação para criar o modelo ideal. Não é bom para um cientista de dados que ele fique trabalhando por semanas ou meses na resolução de um problema e na hora que fez todo o processo de acordo com seu entendimento, os solicitadores dizem que não era aquilo. Para evitar este problema, os chamados *Baby Steps* auxiliam para que as entregas sejam menores e os solicitantes tenham capacidade de acompanhar passo a passo a evolução do projeto. Isso funciona para este processo, pois como as atividades são cíclicas depois de algumas iterações volta para a atividade de identificação do problema e a conversa com a área de negócios.

**Passo 2 - Coleta e limpeza de dados:** A coleta dos dados é a primeira parte deste processo e garante que o dado correto seja adquirido de forma

eficiente. Estes dados brutos podem ser oriundos das bases de dados da empresa, dados de sensores de telemetria, dados originais de serviços de *marketplace*, e tantos outros. O segundo passo é limpar os dados para que a criação do modelo seja efetiva. Limpar os dados significa trabalhar com valores inexistentes, remover os *outliers* (lembra do *outlier* no diagrama de caixas apresentado na “categoria quatro” anteriormente no capítulo?). As correlações entre as variáveis pode ajudar a distinguir quais elementos são realmente úteis para a análise. Se existem duas variáveis que apresentam valores iguais, não são necessárias as duas na análise, visto que existe uma correlação muito forte entre elas, e ao usar apenas uma no modelo, a resposta pode ser encontrada.

**Passo 3 – Desenvolvimento do Modelo Preditivo:** Esta é a parte que mais gostamos de fazer, pois é onde podemos mostrar todo nosso conhecimento e enfrentamos o grande desafio de encontrar o algoritmo correto para resolver o problema apresentado no passo 1 pela área de negócios. Este é um processo interativo, que nos permite analisar algumas possibilidades existente de algoritmos e comparar o desempenho de cada uma delas em cima dos dados que já estão prontos a partir do passo 2. Após comparar os resultados encontra-se o modelo mais apropriado para aquela amostra de dados.

**Passo 4 – Publicação do Modelo:** Depois de desenvolver o modelo, a publicação do algoritmo para uso com dados reais, e não mais dados de treino e validação, é feita. Neste momento o resultado esperado é que os dados reais encontrem retornos como foram encontrados nos dados testados e isso mostra o quanto nosso algoritmo foi bem desenvolvido e ajustado para resolver os problemas reais que a área de negócios pode enfrentar.

**Passo 5 – Monitoramento da performance:** Como já informado anteriormente, o processo é cíclico e não se encerra após a publicação do modelo. As ações são baseadas em métodos estatísticos e matemáticos e

terão melhorias a cada vez que mais dados forem inseridos. É grande a chance de um processo não ter a melhor resposta nas primeiras vezes que for usado com dados reais, por isso, o processo volta alguns passos para que nós, cientistas de dados, possamos ajustar o algoritmo de forma que ele responda corretamente.

Falando especificamente sobre técnicas de análise preditiva, muitas soluções podem ser usadas como as citadas neste capítulo, que não estão associadas a uma tecnologia específica, e sim ao conceito da técnica apresentada. Algumas destas soluções são gratuitas, outras pagas. Este livro aborda especificamente o uso do Azure Machine Learning para criar análise preditiva e permitir um próximo passo nas análises de seu negócio. Os exemplos de onde utilizar e como ter um benefício no cenário proposto serão feitos utilizando o Azure Machine Learning e as explicações detalhadas de como funciona a técnica utilizada serão feitas nos respectivos capítulos.

## 2 – Introdução à Aprendizagem de Máquina e ao Azure Machine Learning

### O que é Aprendizagem de Máquina?

Aprendizagem de máquina, do inglês, *Machine Learning*, não é uma área nova ou recente. Na verdade, os algoritmos e processos utilizados datam de vários anos atrás. Sendo que existem diversas definições para a área.

Segundo a Wikipedia<sup>3</sup>: "A aprendizagem automática é um subcampo da inteligência artificial dedicado ao desenvolvimento de algoritmos e técnicas que permitam ao computador aprender, isto é, que permitam ao computador aperfeiçoar seu desempenho em alguma tarefa. Enquanto que na Inteligência Artificial existem dois tipos de raciocínio – o indutivo, que extrai regras e padrões de grandes conjuntos de dados, e o dedutivo – o aprendizado de máquina só se preocupa com o indutivo.

Algumas partes da aprendizagem automática estão intimamente ligadas à mineração de dados e estatística. Sua pesquisa foca nas propriedades dos métodos estatísticos, assim como sua complexidade computacional. Sua aplicação prática inclui o processamento de linguagem natural, motores de busca, diagnósticos médicos, bioinformática, reconhecimento de fala, reconhecimento de escrita, visão computacional e locomoção de robôs.".

Outra definição bastante utilizada é a de Arthur Samuel, definida em 1959, que define como: *um campo de estudo que permite aos computadores aprenderem sem serem explicitamente programados.*

E por fim, uma das definições mais formais é a fornecida por Tom M. Mitchell: *Um programa “aprende” a partir de experiências “E” com respeito a alguma classe de tarefas “T” e medida de desempenho “P”, se o desempenho de tarefas em “T”, medido por “P”, melhora com experiências “E”.*

---

<sup>3</sup> [https://pt.wikipedia.org/wiki/Aprendizado\\_de\\_m%C3%A1quina](https://pt.wikipedia.org/wiki/Aprendizado_de_m%C3%A1quina)

É importante notar que não é por existirem várias definições que uma invalida a outra. Particularmente, acreditamos que aprendizagem de máquina é a capacidade de dar subsídios para um programa computacional e obter respostas que de forma manual poderiam ser impossíveis, dado a complexidade, volume de dados e a alta dimensionalidade tratadas nos problemas. O objetivo de um sistema destes é obter uma solução (em geral um modelo ou a associação de vários modelos) para responder à problemas reais, tais como: previsão do tempo, detecção de formas diferentes de câncer, análise de perfil de compradores, análise de fala/escrita, entre muitas outras aplicações.

Uma outra visão sobre o que é *machine learning* pode ser encontrada (em inglês) no blog de Machine Learning da Microsoft<sup>4</sup>, escrita por John Plat (*Distinguished Scientist* no Microsoft Research).

## As Motivações para Aprender

Caso apareça a dúvida sobre quando utilizar (ou não utilizar) um modelo de aprendizagem, pense nos seguintes cenários sobre quando devemos utilizar:

Aprenda quando você não pode codificar. Podemos citar: reconhecimento de fala, imagens e gestos.

Aprenda quando você não pode escalar. Podemos citar: recomendações, classificação de e-mail indesejados (spam), e detecção de fraudes.

Aprenda quando você precise adaptar e/ou personalizar. Podemos citar: escrita preditiva.

---

<sup>4</sup> <https://blogs.technet.microsoft.com/machinelearning/2014/07/01/what-is-machine-learning/>

Aprenda quando você não pode fazer o acompanhamento manual. Podemos citar: inteligência artificial em jogos e controle de robôs.

## Conceitos Fundamentais

Já sabemos como é o dia a dia de um cientista de dados e também vimos algumas definições do que é aprendizagem de máquina. Para o restante do livro fazer sentido e para que você possa entender de forma completa sobre os tópicos abordados, precisamos introduzir alguns conceitos iniciais que são a base para os processos que veremos.

### Características

Características, do inglês, *features*, são os componentes básicos que definem o objeto de estudo, ou seja, um conjunto de características é o que descreve/representa um objeto. A extração/seleção de características não é uma tarefa fácil, e frequentemente, pode se tornar na atividade mais onerosa de todo o processo. Uma característica efetiva, em geral, deveria ser invariante a escala ou rotação de um dado objeto de estudo. Estas características podem ser Categóricas ou Numéricas, esta última se subdividindo em Discretas ou Contínuas.

Por exemplo, imagine um problema onde temos um conjunto de fotos de animais e queremos identificar a espécie presente em cada foto através de um algoritmo de aprendizagem de máquina. Devemos ter características invariantes a escala e rotação: um cachorro continua sendo um cachorro, seja uma foto grande ou pequena, esteja de frente ou de lado. É claro que modelos capazes de identificar vários cenários são em geral bastante complexos.

Características Categóricas são aquelas que atuam como uma categoria, e normalmente possuem um conjunto conhecido e não mensurável (calculável) de valores. Exemplos nesta categoria no nosso contexto acima poderiam ser: cor do pelo, presença de asas ("sim"/"não"), presença de penas ("sim"/"não"), entre diversas outras. Em geral são valores não numéricos (palavras), mas isso não pode ser tomado como regra pois um número pode

ser categórico (representar uma categoria). Muitos algoritmos necessitam de apenas números como padrões de entrada, então é comum converter palavras em números e marca-los explicitamente como características categóricas.

Características Numéricas são aquelas que possuem um conjunto possivelmente infinito de valores diferentes. Em geral valores numéricos. Exemplos de características, neste contexto, para nosso exemplo poderiam ser: peso do animal; cumprimento do animal; entre outros. Estes valores são considerados contínuos pois possuem uma variedade não limitada de possibilidades, por exemplo, o peso, que pode ser medido em uma balança de precisão e conseguir diferenciar alguns miligramas entre cada animal. Já os valores numéricos discretos são possibilidades conhecidas do intervalo possível de valores, por exemplo, quantidade de patas de um animal. As patas normalmente serão sempre entre 0 e 4, divididas em intervalos inteiros conhecidos.

Existem algoritmos que se adaptam melhor a valores categóricos (exemplo: árvores de decisão, do inglês, *decision trees*) e existem aqueles que se adequam melhor a valores numéricos (exemplo: máquinas de vetor de suporte, do inglês, *support vector machines*).

Uma técnica que é bastante utilizada sobre as características contínuas para adaptar em algoritmos não favoráveis a elas é a discretização. Por exemplo, ao invés de ter uma característica do peso do animal (contínua), poderíamos ter uma dividida em grupos: "[0 – 100g]"; "[100g – 1Kg]"; "[1Kg – 5 Kg]"; "[Mais de 5Kg]". Claro que estes intervalos devem ser definidos de acordo com a necessidade do problema a ser resolvido.

## Classes

Uma classe, também conhecida como rótulo, do inglês, *label*, é o valor que define o objeto.

Lembra nosso problema apresentado na sessão de características sobre criar um modelo para identificação de um animal? O conjunto de classes neste problema é enorme, pois seria possivelmente todas as espécies de animais existentes. Um problema destes é bastante complexo, pois necessitamos informações e exemplos de muitos animais diferentes.

Os problemas na vida real em geral tratam um conjunto bem menor de possíveis classes: duas ou poucas classes é o comum. Problemas com muitas classes, principalmente os a serem resolvidos com algoritmos de aprendizagem supervisionada, exigem uma base de dados grande e que conte com exemplos de todas as classes desejadas. Em geral é melhor criarmos diversos modelos específicos do que tentar um modelo genérico que trata de muitos casos.

Por exemplo, imagine que o nosso problema seja a detecção de câncer. Podemos ter um modelo que identifica qualquer tipo de câncer, que provavelmente seria inviável de ser construído, ou podemos ter diversos modelos que tentam identificar a presença de um tipo específico de câncer (possui ou não possui câncer de mama, possui ou não possui câncer de pele, etc.). Problemas de duas classes em geral são mais simples que problemas de três classes e assim por diante.

O formato da resposta desejado ao problema é o que determinará a quantidade de classes, algoritmos a serem usados e também a complexidade do modelo.

## Aprendizagem Supervisionada e Não-Supervisionada

Os algoritmos de aprendizagem podem ser agrupados/classificados segundo a sua abordagem, sendo os dois maiores grupos a aprendizagem supervisionada e aprendizagem não-supervisionada. Não abordaremos neste livro, mas existem outras abordagens como: aprendizagem por reforço, aprendizado multitarefa, aprendizagem semi-supervisionada, entre outras.

Um ponto é comum em todos os modelos: Para operarem de forma satisfatória necessitam de dados sobre o problema a ser tratado. São através destes dados que ensinamos os modelos sobre o problema em questão. A atividade de ensinar um modelo é o que chamamos de treinamento.

Problemas supervisionados são aqueles que necessitam conhecer os elementos (suas características), bem como suas classes. Em outras palavras, o conjunto de treino precisa estar rotulado.

Um conjunto de treino é normalmente representado um conjunto de pares, onde cada par é composto normalmente de um vetor de características e o objeto de saída desejado (classe).

Para problemas mais complexos e/ou com muitas classes são necessários conjuntos de treino maiores (com representatividade em todas as classes estudadas). A seguir podemos observar um exemplo de um conjunto de dados que pode ser utilizado. Note que há apenas 4 exemplos, no mundo real para problemas mais complexos podemos ter milhares de exemplos (ou até mais).

```
Class, sepal-length, sepal-width, petal-length, petal-width
1, 6.3, 2.9, 5.6, 1.8
0, 4.8, 3.4, 1.6, 0.2
1, 7.2, 3.2, 6, 1.8
0, 5.2, 3.4, 1.4, 0.2
```

Em destaque temos a classe de cada exemplo. Modelos que utilizam aprendizagem supervisionada precisam ser semelhantes a este. A informação da classe correta é utilizada pelos algoritmos durante o treino para ajustar o seu erro e assim maximizar a acurácia. Também utilizamos a classe, que conhecemos de antemão ao treino, para verificar o quanto assertivo é o mesmo, através da comparação da classe correta com a classe que foi prevista com o modelo.

Podemos citar alguns algoritmos que se encaixam na aprendizagem supervisionada: máquinas de vetor de suporte (*SVM – support vector machines*), redes neurais (*neural networks*), árvores de decisão (*decision trees*), k-vizinhos mais próximos (*k-NN – k-nearest neighbors*), entre outros.

A aprendizagem de máquina não supervisionada aborda os algoritmos que não necessitam conhecer de antemão a classe correta. Para estes casos não é possível fazer uma comparação final automatizada para verificar o acerto (acurácia), mas existem vários cenários que podem se beneficiar de algoritmos com esta abordagem.

Dentre as atividades que se enquadram melhor na aprendizagem não supervisionada a mais comum é o agrupamento (clusterização) dos exemplos.

Imagine que o desejado é agrupar consumidores semelhantes. Em geral temos como entrada para os algoritmos de clusterização o número de classes (N) que imaginamos existir no conjunto de dados. Com isso, ao término da execução do modelo teremos os exemplos separados em N classes.

Podemos citar alguns algoritmos que se encaixam na aprendizagem não-supervisionada: *k-means*, algoritmo *cocktail party*, entre outros.

Os algoritmos supervisionados são mais numerosos que os não-supervisionados. Na plataforma do Azure Machine Learning é possível encontrar os principais, bem como diversos outros, já implementados e prontos para seus experimentos. Abordaremos os principais algoritmos no Capítulo 7.

## Azure Machine Learning – AzureML

Nos anos 90 e início dos anos 2000 os modelos de aprendizagem de máquina envolvendo grandes massas de dados poderiam passar diversas horas sendo processados, e em alguns casos dependendo do volume e complexidade do problema poderiam levar vários dias.

A comparação do poder de computação dos anos 90 com hoje é injusto, mas hoje temos volumes de dados cada vez maiores (além de problemas mais complexos) e lidar com estes dados para construir modelos usando um desktop ou laptop se torna cada vez mais desafiador. É possível investir e comprar servidores para realizar estas tarefas, mas em geral isso envolve custos bastante elevados.

Os processos, técnicas e algoritmos da área consomem bastante poder computacional (elevada utilização de memória RAM e CPU, e em alguns casos até GPU) durante a criação de seus modelos. Como muito da área é baseado em experimentação, diferentes parâmetros e algoritmos, diferentes

subconjuntos de dados e técnicas de combinação, torna-se necessário diversas (dezenas ou centenas) execuções até chegar em um modelo viável.

Com isto em mente e com a evolução da capacidade e dos serviços de computação em nuvem a Microsoft criou uma plataforma baseada em nuvem para aprendizagem de máquina batizada de *Azure Machine Learning*.

Imagine que com esta tecnologia é possível fazer a experimentação e construção dos modelos sem a necessidade de pagar pela aquisição e manutenção de grandes e caros servidores. O pagamento é mediante uso. Existe inclusive uma versão gratuita do AzureML para teste da plataforma.

O Azure Machine Learning (AzureML<sup>5</sup>) é uma plataforma baseada na web , ou seja, não há a necessidade de instalar nenhum software específico no computador do cientista de dados, e o mesmo pode continuar o seu trabalho de qualquer estação conectada à internet. Conforme veremos nos capítulos a seguir, as possibilidades na criação dos modelos são bastante expressivas, e a sua capacidade de customização (seja com código escrito em *R* ou *Python*) tornam o AzureML bastante atrativo.

Os modelos construídos e executados na plataforma são executados sobre servidores escaláveis e com grande capacidade de processamento, o que permite treinar um modelo (veja mais sobre treino no Capítulo 4) complexo em minutos. Isso faz com que o cientista de dados possa focar no seu trabalho sem se preocupar com servidores, conexões, infraestrutura, desempenho, entre outros aspectos.

Além da construção dos modelos serem realizadas através de um navegador, o consumo destes modelos também é através do consumo de um serviço de nuvem. Suas APIs são baseadas em *web services*, o que facilita a sua utilização/integração em qualquer linguagem de programação moderna. Para testes é possível utilizar uma interface no Excel ou através do próprio AzureML Studio, conforme veremos no Capítulo 6.

---

*5 Site do Azure Machine Learning: <http://studio.azureml.net>*

## Conhecendo o AzureML Studio

A interface para criação dos modelos de *machine learning* é chamada de Studio. Ela é possível de ser acessada através do AzureML<sup>6</sup>, retratada pela Figura 17.

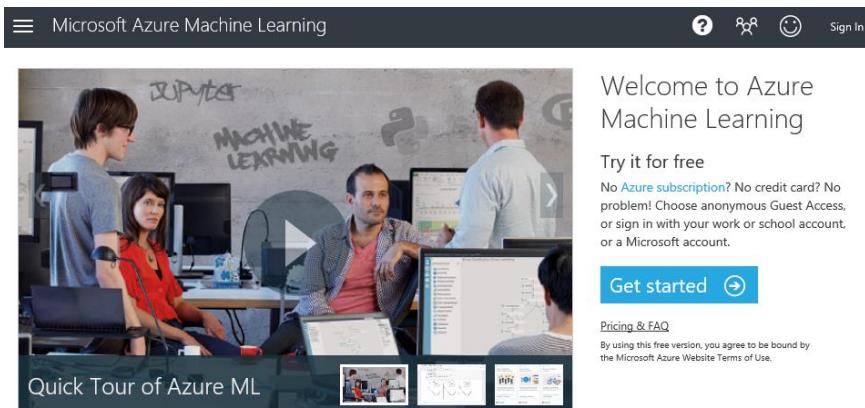


Figura 17 - Portal de entrada

Para os que desejam testar e conhecer a plataforma (e para a maior parte dos exemplos mostrados neste livro) é possível utilizar o AzureML de graça e sem a necessidade de cartão de crédito, basta apenas uma conta Microsoft (Live, Hotmail, Outlook, etc.).

Após entrar com as suas credenciais é possível observar o portal representado pela Figura 18.

*Nota: É importante ressaltar que a plataforma recebe atualizações constantes (aproximadamente a cada 3 meses): novos algoritmos, novas formas de facilitar a vida do cientista de dados, bem como mudanças na interface gráfica (principalmente mudança nos menus). Então não se preocupe se a sua interface estiver ligeiramente*

<sup>6</sup> Site do Azure Machine Learning: <http://studio.azureml.net>

*diferente da apresentada, pois os conceitos para a sua utilização e o que você irá aprender com este livro não será prejudicado.*

	NAME	AUTHOR	STATUS	PROJECT
Sample [Pr...]	zavaschi	Draft	1.. None	
Retrain	zavaschi	Finished	1.. None	
Sample [Pr...]	zavaschi	Finished	1.. None	
Experiment...	zavaschi	Canceled	1.. None	
Anomaly D...	AzureML Te...	Finished	1.. None	
Anomaly D...	zavaschi	Draft	1.. None	
Experiment...	zavaschi	Finished	9.. None	
R - ggdendro	zavaschi	Finished	7.. None	
Experiment...	zavaschi	Finished	7.. None	
R - 3	zavaschi	Finished	7.. None	
TDC - Com...	zavaschi	Finished	7.. None	
TDC - Com...	zavaschi	Finished	7.. None	

Figura 18 - Portal do AzureML

Existem 7 abas na região esquerda do ML Studio: Projects, Experiments, Web Services, Notebooks, Datasets, Trained Models e Settings.

### Aba Projects

Com a utilização da plataforma, é normal o número de experimentos, conjuntos de dados e demais objetos crescerem bastante. Isso tornava a organização uma tarefa complicada, principalmente ao trabalhar em experimentos distintos em um mesmo espaço de trabalho (*workspace*).

A essência de um projeto é agrupar estes diferentes objetos para facilitar a sua organização e controle. Esta aba possibilita a criação e manutenção de projetos.

### Aba Experiments

Esta região se subdivide em duas abas internas: *My Experiments* (meus experimentos) e *Samples* (exemplos). Esta aba pode ser observada na Figura 2.

Na parte de *My Experiments* encontram-se todos os experimentos criados (ou copiados de outros espaços de trabalho) nesta conta, bem como uma visualização do experimento criado.

Já em *Samples* existe uma grande coleção de experimentos prontos. Estes foram desenvolvidos pela Microsoft ou outros fornecedores e podem ser usados como referência. Existem alguns com objetivo de tutorial, mas existem dezenas deles retratando problemas do mundo real.

Estes exemplos também foram expandidos para uma galeria própria<sup>7</sup> (sob a bandeira das soluções inteligentes da Microsoft – *Cortana Intelligence Suite*) para expor e compartilhar experimentos. Esta galeria pode receber contribuições de qualquer pessoa que esteja desenvolvendo com a plataforma do *Azure Machine Learning*.

## Aba Web Services

Nesta aba é possível observar os modelos expostos através da plataforma. A forma de comunicação das aplicações com o serviço do AzureML é através de *web services*. É possível fazer a configuração destes web services nesta região também. Os *web services* do AzureML são abordados com maior detalhamento no Capítulo 6.

## Aba Notebooks

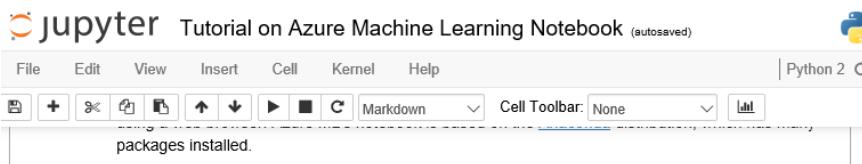
Esta aba permite a visualização de todos os *notebooks* já criados. No momento da escrita deste livro o recurso se encontra ainda em *preview* (visualização) e permite a interação e criação de *Jupyter notebooks* para as linguagens *Python* e *R*.

---

<sup>7</sup> Galeria do Cortana: <https://gallery.cortanaintelligence.com>

Através dos notebooks o cientista de dados pode criar documentos vivos para documentar o processo que realizou, mostrar a sua lógica de pensamento, baseado numa plataforma online que pode armazenar texto ou código, bem como executar este código e já adicioná-lo dentro do documento.

Para entender melhor o funcionamento do Azure ML Notebook a Microsoft criou um notebook de exemplo que pode ser criado através do botão + NEW -> Notebook -> *Tutorial on Azure Machine Learning Notebook*, acompanhe a Figura 19.



## 2 Data

In this example we'll be using the Boston housing dataset. There are 506 rows in the dataset. The target variable is median home price. There are 13 predictor variables including average number of rooms per dwelling, crime rate by town, etc. More information about this dataset can be found at [UCI](#).

```
In [1]: from sklearn.datasets import load_boston
boston = load_boston()
X = boston.data
y = boston.target
feature_names = boston.feature_names
print(X.shape)
print(y.shape)
print(feature_names)

(506, 13)
(506,)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT' 'MEDV']
```

Figura 19 - Jupyter Notebook utilizando linguagem Python

## Aba Datasets

Nesta aba é possível fazer a gestão dos conjuntos de dados que foram utilizados (*upload*) nos experimentos criados ou que ainda serão utilizados. Também há a divisão em duas abas internas: uma para os seus *datasets* e outra

para os utilizados nos experimentos de exemplo. Esta aba pode ser observada na Figura 20.

Os *datasets* de exemplo (e os seus) podem ser reutilizados em quantos experimentos desejar, é uma excelente forma de iniciar o estudo da plataforma sem ter que montar bases de dados para análise.

NAME	SUBMITTED BY	DESCRIPTION	DATA TYPE	CREATED	SIZE	PROJECT
BlindSet_Income	zavaschi		Dataset	9/16/2014 10:08:5...	978.22 KB	None
Zoo dataset	zavaschi		GenericCSVNoHe...	1/22/2015 3:37:02...	4.13 KB	None
NOAA Weather Ju...	zavaschi		GenericCSV	1/22/2015 5:08:30...	22.59 MB	None
myRScript.zip	zavaschi		Zip	1/22/2015 8:11:51...	150 B	None
Boston.zip	zavaschi		Zip	1/22/2015 8:25:53...	14.76 KB	None
ggdendro_0.1-14...	zavaschi		Zip	1/22/2015 8:33:48...	116.13 KB	None
Boston housing d...	zavaschi		Dataset	1/22/2015 8:35:58...	25.12 KB	None
CRM Dataset Shar...	weehyong tok	CRM Dataset	GenericTSV	6/23/2015 3:50:44...	24.99 MB	None
CRM Churn Label...	weehyong tok	CRM Churn Labels	GenericCSVNoHe...	6/23/2015 3:51:01...	191.73 KB	None
Alcohol Consump...	Jaganadhi Gopina...	Alcohol Consum...	GenericCSV	9/23/2015 7:45:59...	1.32 KB	None

Figura 20 - Aba de Datasets

Algo que veremos na parte de integração com R (Capítulo 9) é a capacidade da utilização de bibliotecas, scripts próprios, e arquivos de dados do R (.RData) na plataforma. Para tal podemos fazer upload de um arquivo .zip contendo estes arquivos. Estes arquivos .zip podem ser vistos e gerenciados nesta aba.

Também é possível apagar ou fazer *download* destes datasets, abrir em um notebook, ou gerar um código de liberação para acesso programático ao *dataset* da IDE de desenvolvimento da sua escolha (sem necessariamente ser através do AzureML). Esta opção pode ser observada na Figura 21. É importante perceber que o *token* presente no código concede acesso a todo

seu *workspace*. Trate ele com as mesmas restrições e segurança que uma senha teria.

×

GENERATE DATA ACCESS CODE

Use this code to access your data

To programmatically access this dataset, simply copy the code snippet into your favorite development environment. [Learn More](#).

Note: this code includes your workspace access token, which provides full access to your workspace. It should be treated like a password.

CODE SNIPPET

Python

```
from azureml import Workspace
ws = Workspace(
    workspace_id='[REDACTED]',
    authorization_token='[REDACTED]',
    endpoint='https://studioapi.azureml.net'
)
ds = ws.datasets['Blood donation data']
frame = ds.to_dataframe()
```

USE SECONDARY TOKEN

(checkmark)

Figura 21 - Código para acessar um dataset programaticamente

## Aba Trained Models

Ao terminar um experimento (normalmente após treiná-lo) é possível armazenar este modelo para sua utilização em novos experimentos. Esta aba armazena e permite a gestão destes modelos salvos.

## Aba Settings

Esta aba contém três sub abas: *Name*, *Authorization Tokens* e *Users*.

Na parte de *Name* (Figura 22) é possível definir o nome do *workspace*, visualizar seu tipo (se é modelo gratuito<sup>8</sup>) e identificar quanto do espaço está sendo consumido. Por exemplo, para uma assinatura gratuita podemos armazenar até 10 GB de dados.

Na parte de *Authorization Tokens* podemos ver os *tokens* de acesso ao *workspace*. Rate os com cuidado pois com eles é possível fazer acesso aos dados de seu *workspace*.

Por fim em *Users* podemos compartilhar nosso *workspace* com outras pessoas (basta apenas informar a conta que utilizam). Os usuários terão acesso a todos as informações de seu *workspace* (experimentos, etc.).

The screenshot shows the 'settings' tab of the Microsoft Azure Machine Learning interface. On the left, there's a sidebar with icons for EXPERIMENTS, WEB SERVICES, NOTEBOOKS, DATASETS, TRAINED MODELS, and SETTINGS. The SETTINGS icon is highlighted. The main area has tabs for NAME, AUTHORIZATION TOKENS, and USERS, with NAME selected. Under NAME, there are fields for WORKSPACE NAME (containing 'Thiago Zavaschi-Free-Workspace'), WORKSPACE DESCRIPTION (containing 'Default workspace.'), WORKSPACE TYPE (set to 'Free'), WORKSPACE ID (containing '780855a4efec4e52af768fd5195f96f9'), and WORKSPACE STORAGE (showing '0 GB' used of 10 GB available). A progress bar at the bottom indicates 0% of 10 GIGABYTES used.

Figura 22 - Aba Settings - Name

8 <https://azure.microsoft.com/en-us/pricing/details/machine-learning/>

## Botão + NEW

O botão + *NEW* abre a opção para criação de um novo experimento, *module* (zip), *dataset* ou um *notebook*, conforme pode ser visto na Figura 23.

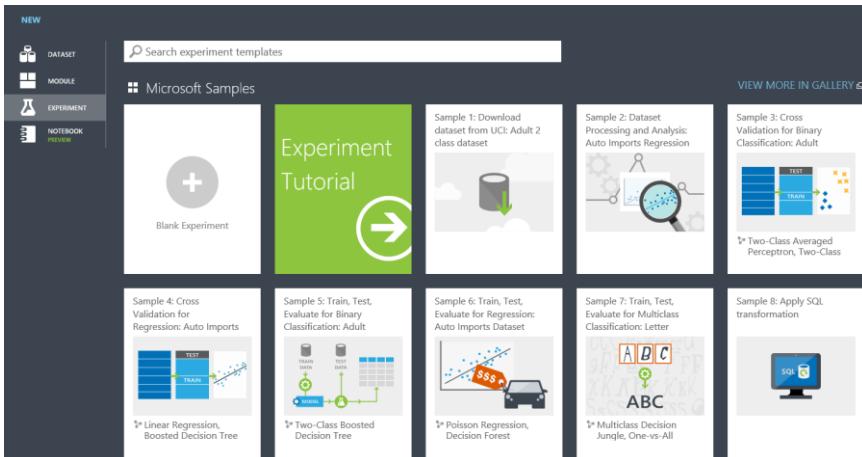


Figura 23 - Opções do botão + New

A opção de *Dataset* serve para fazer o upload de um arquivo local que pode ser dos seguintes formatos:

- *Generic CSV file with a header* (.csv)
- *Generic CSV file with no header* (.csv)
- *Generic TSV file with a header* (.tsv)
- *Generic TSV file with no header* (.nh.tsv)
- *Plain text* (.txt)
- *SvmLight file* (.svmlight)
- *Attribute Relation file format* (.arff)
- *Zip* (.zip)

- *R Object or Workspace (.RData)*

Este *dataset* se torna disponível na área de *Saved Datasets* (ver *Aba Datasets*) que é armazenada em blobs do Microsoft Azure.

## Módulos do Azure Machine Learning

Os processos e elementos utilizados dentro de um experimento (além dos dados) são chamados de módulos.

Estes módulos são utilizados para compor o fluxo de um experimento: pré-processamento, aquisição de dados, treino, validação, algoritmos, entre muitos outros. Estes módulos já contêm as boas práticas adquiridas dos mais de 20 anos de uso de aprendizagem de máquina pela Microsoft.

A Figura 24 retrata a estrutura de módulos disponíveis (categorias e subcategorias) no AzureML.

- ▶  Saved Datasets
- ▶  Data Format Conversions
- ▶  Data Input and Output
- ◀  **Data Transformation**
  - ▶  Filter
  - ▶  Learning with Counts
  - ▶  Manipulation
  - ▶  Sample and Split
  - ▶  Scale and Reduce
- ▶  Feature Selection
- ◀  **Machine Learning**
  - ▶  Evaluate
  - ◀  Initialize Model
    - ▶  Anomaly Detection
    - ▶  Classification
    - ▶  Clustering
    - ▶  Regression
    - ▶  Score
    - ▶  Train
- ▶  OpenCV Library Modules
- ▶  Python Language Modules
- ▶  R Language Modules
- ▶  Statistical Functions
- ▶  Text Analytics
- ▶  Web Service
- ▶  Deprecated

Figura 24 - Categorias e subcategorias de módulos do AzureML

Ao longo dos capítulos do presente livro iremos abordar os principais módulos, bem como suas diferenças e informações sobre como utilizá-los.

## 3 – Trabalhando com Dados Externos

### A importância dos dados

Você sabe fazer as perguntas certas para seus dados? Começar com esta provocação que coloca em dúvida a nossa capacidade de obter as respostas necessárias com os dados que estão acessíveis faz com que nós pensemos em quais perguntas devemos fazer para os dados antes de começar a desenvolver nossos modelos e soluções. Muitas vezes os solicitantes fazem pedidos sem pensar se a pergunta é clara o suficiente e se existem dados para dar apoio e responder a tais perguntas.

Imagine o seguinte cenário: você é gerente de marketing de uma empresa que vende serviços pela internet. Seu bônus no fim do ano está atrelado ao aumento das vendas destes serviços. Você contrata um ótimo desenvolvedor de software e pergunta pra ele: *"Como faço pra vender mais?"*.

Refletindo sobre esta pergunta, pode-se entender que ela é ampla e mal direcionada. Isso não é um erro propriamente dito, muitas vezes os nossos solicitantes não sabem fazer uma pergunta de forma direcionada por não fazerem parte de uma cultura de dados. Este mal direcionamento na pergunta pode provocar uma execução da atividade ruim.

Algumas das ações possíveis para este desenvolver poderiam incluir direcionar toda sua energia para melhorar a usabilidade do sistema, melhorar o tempo de resposta para uma solicitação, e diversas outras ações focadas no código da aplicação que ele pode inferir como impactantes para a venda. Porém, se você, gerente de marketing, faz a mesma pergunta num outro sentido como por exemplo "Como faço para um usuário que colocou um produto no carrinho virtual, efetivar de fato a compra?", será que o resultado do esforço do desenvolvedor seria diferente também? Repare na diferença da pergunta, embora o objetivo final seja o mesmo (vender mais), o desenvolvedor foi direcionado a resolver um problema existente e não "caçar" um problema que não sabe se existe.

Neste momento, há chance deste desenvolvedor de software separar o universo de clientes em dois grupos, sendo, um grupo com os usuários que efetivamente fizeram a compra e no outro grupo os usuários que colocaram o serviço no carrinho virtual, mas não o compraram. O desenvolvedor provavelmente vai se esforçar para encontrar a diferença do comportamento destes dois grupos e então poderá criar intervenções no grupo que só coloca os produtos no carrinho para que eles façam as mesmas coisas que o outro grupo fez. Chegando no resultado esperado, que é a empresa vender mais.

Agora “troque o chapéu” e assuma a posição do desenvolvedor de software. Você foi contratado por um gerente de marketing de uma empresa que vende serviços pela internet. Este gerente te pediu: “Como faço para vender mais?”. Neste momento seu instinto de cientista de dados conduz uma conversa com o gerente e comprehende com detalhes o que ele entende por “vender mais”. Você acaba descobrindo que a real pergunta é “Como faço pra um usuário que colocou o produto no carrinho, fazer o pagamento?”. Com base nestas conversas você direciona os esforços para algumas soluções que atenderão a necessidade do gerente, não necessariamente olhar o código fonte da aplicação em busca de um problema que à princípio não existe ali. Seu esforço será trabalhar em cima dos dados e entender qual é o público que mais compra, quais campanhas promocionais possuem melhor desempenho, quais canais de publicidade tem retorno sobre investimento melhor, entre outros aspectos que obteve de ideias durante a conversa. A sua curiosidade em aplicar modelos matemáticos e estatísticos nos dados atrás de conseguir melhores resultados será importante para obter sucesso nas respostas. Depois de algumas análises e cruzamentos é possível contar uma história de forma eficiente através dos dados, provando matematicamente as chances de aumento nas vendas caso algumas ações forem realizadas.

Você como cientista de dados contratado para resolver este problema poderia seguir por alguns caminhos, como por exemplo, entender através dos dados do grupo que efetiva as compras qual o percentual de abertura dos e-mails de promoção e quantos usuários não abrem. Pense em uma amostra de dados dividida conforme a tabela a seguir.

Grupo 1 – Compra	Grupo 2 – Carrinho	Total
1.500 pessoas	7.000 pessoas	8.500 pessoas

Avançando neste raciocínio, imagine segmentar estes dois grupos existentes, em quatro grupos. Onde cada um dos grupos é subdividido em dois novos grupos separando os usuários que abriram os e-mails dos usuários que não abriram. O resultado desta nova segmentação, baseada na abertura de e-mails, poderia ser similar ao representado na tabela a seguir.

Grupo 1 – Compra		Grupo 2 – Carrinho		Total	
	1.500 pessoas		7.000 pessoas		8.500 pessoas
Abriram	Não Abriram	Abriram	Não Abriram	Abriram	Não Abriram
1.000	500	1.760	5.240	2.760	5.740

Ao olhar estes dados, é possível ver uma causalidade existente entre os usuários que compram e que não compram ligada à taxa de abertura de e-mails. Estes dados nos mostram que muitas pessoas que compraram o produto abriram os e-mails, e das pessoas que não compraram a taxa de abertura é menor. Isso direciona o resultado para que, ao fazer mais usuários do grupo 2 abrir e-mails, eles tendem a comprar o serviço. Não é certeza que esta formula vá funcionar para todos os casos, mas é uma hipótese suportada por dados. Vale lembrar aqui que estes números são fictícios e foram usados apenas para ilustrar a explicação.

Depois de entender brevemente que é preciso fazer a pergunta correta para os dados, pode surgir a dúvida sobre quais dados ou quais tipos de dados estão disponíveis para ajudar a responder às questões. Basicamente existem dois tipos de dados, os dados estruturados e os não estruturados. Algumas literaturas chegam a definir dados semiestruturados, mas estes são conceitualmente reconhecidos como um híbrido dos dados estruturados e não estruturados, então não será abordado neste livro.

Em linhas gerais, os dados estruturados possuem uma estrutura conhecida pelos envolvidos no desenvolvimento de software que interage com aquele dado, onde sabe-se a posição que inicia o armazenamento de um campo, seu tamanho, tipo de dado que é atribuído àquele pedaço da informação, etc. Trazendo este exemplo para um mundo de bancos de dados, pode-se imaginar um campo existente dentro de uma tabela, este campo é

do tipo numérico e inteiro. Isso garante que é conhecido seu ponto inicial, ponto final, e somente terá caracteres que sejam entre 0 e 9. Qualquer outro caractere que estiver dentro deste campo é errado e espera-se que o Sistema Gerenciador de Banco de Dados apresente um erro sempre que alguém tentar informar um caractere diferente do que é aceito pelas definições. Este campo numérico inteiro pode fazer parte de uma estrutura mais ampla, como uma tabela. Avançando um pouco mais no exemplo, imagine uma estrutura para armazenar os dados individuais de uma pessoa. Uma forma de modelar este armazenamento pode ser os seguintes campos/atributos.

Campo	Tipo	Tamanho (posições)
ID	Inteiro	8
Nome	Texto	50
DataNascimento	Data	8
Sexo	Caractér	1

Se esta representação de registro de pessoa for armazenada em um arquivo de texto que mantém os dados, respeitando o tamanho informado de cada campo e o tipo de dado, não muda entre um registro e outro, pode-se dizer que se trata de um tipo de dado estruturado. Sempre que precisar retornar a data de nascimento de uma pessoa, basta recuperar os caracteres existentes entre a posição 59 e 65. A data de nascimento começa na posição 59, pois é a primeira posição após os oito caracteres do ID e os cinquenta caracteres do Nome.

Alguns exemplos de dados estruturados podem ser encontrados em SGBDs (sistemas gerenciadores de bancos de dados), arquivos CSVs (comma separated value) e suas variações, XMLs (extensible markup language) e JSONs (javascript object notation), e diversos outros.

Ao trabalhar com dados não estruturados, o principal empecilho é descobrir onde está o pedaço da informação dentro daquela massa de dados que possui para trabalhar.

Imagine que você está em um desenvolvimento de um sistema que, a partir de uma fotografia, identifica indivíduos que usam óculos e outros que não usam. Não se sabe ao certo qual parte da fotografia existe este dado, que é o indivíduo utilizando óculos. Uma técnica para fazer isso, pode ser treinar uma rede neural com diversos exemplos de óculos e pessoas que utilizam óculos, depois de treinar esta rede neural, é preciso validar a foto com base neste modelo que possui a caracterização dos exemplos de óculos. Repare que não se sabe se existe um óculos na imagem, ou caso exista, qual é o ponto exato de sua localização. Neste caso é preciso percorrer toda a estrutura atrás de encontrar o objeto. No caso de imagens, uma forma é dividir a imagem em diversos quadrantes, pode ser de 50 x 50 pixels, e começar a analisar cada um destes quadros isoladamente. Em uma imagem de 250 x 250 pixels, esta divisão gera 5 quadros por linha, dividido em 5 linhas. Veja a Figura 25 a seguir para entender esta divisão.

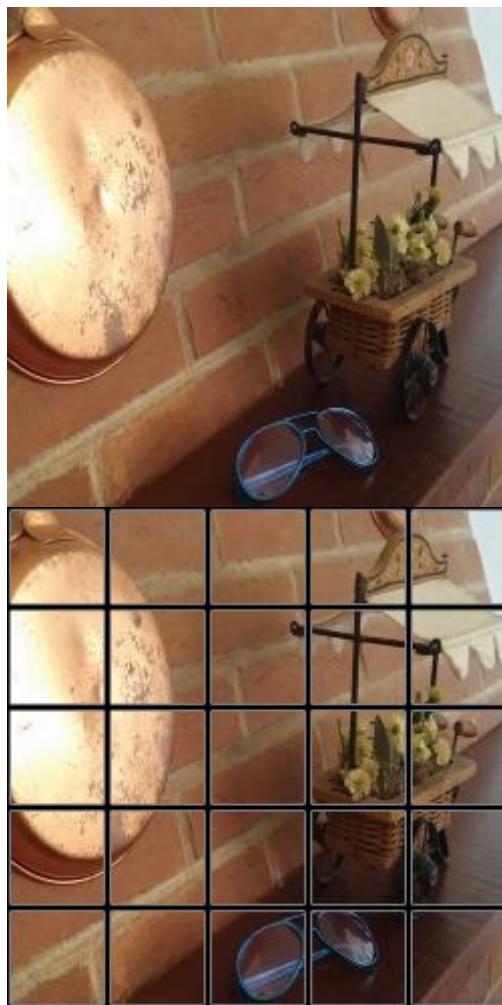


Figura 25 - Divisão da imagem por quadrantes

O processo de busca pelo óculos começa a percorrer o vetor de quadros, e tenta localizar algum objeto de acordo com a semelhança com os objetos que foram treinados, no caso, os óculos. Ao encontrar os objetos que são semelhantes, é marcado como encontrado. No exemplo abordado, o algoritmo aponta que existe um óculos nas posições 5,3 e 5,4 (linha 5 com as

colunas 3 e 4). Apresentados na Figura 26 a seguir, com os quadrantes mais claros:



Figura 26 - Ilustração da localização do objeto buscado

Uma segunda visão da mesma figura (Figura 27), pode receber o mesmo processo utilizado no exemplo anterior.

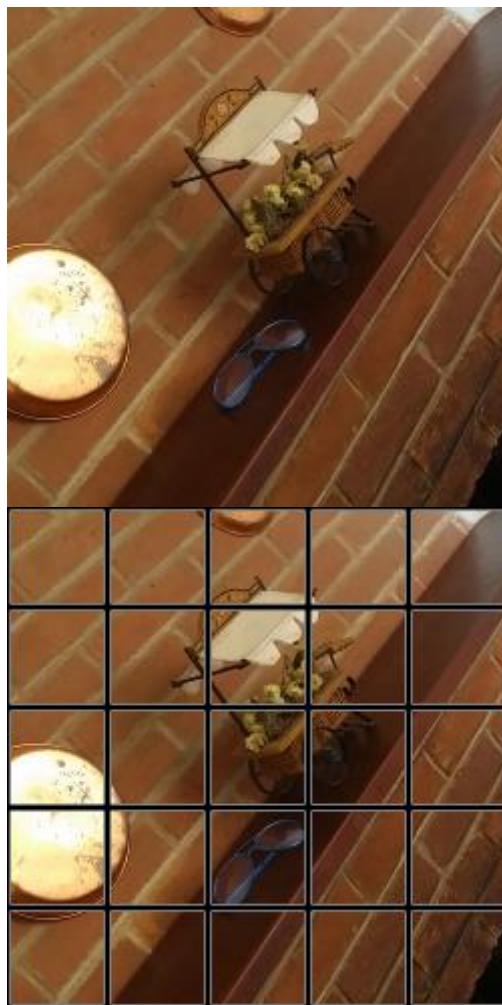


Figura 27 – Exemplo similar com localização diferente do objeto buscado

E o algoritmo encontrará o objeto em um outro lugar, diferente o primeiro, desta vez na posição 4,3 (linha 4 e coluna 3). Conforme pode ser observado na Figura 28.



Figura 28 – Localização do objeto buscado

Isso nos remete ao conceito de dados não estruturados, fazendo com que não seja tão claro o local exato onde cada objeto se encontra dentro da massa de dados que está sendo utilizada como origem. Além de que cada formato (texto, vídeo, imagem) pode exigir uma técnica diferente para localização e análise dos dados.

Exemplos de dados não estruturados podem ser facilmente encontrados ao trabalhar com fotos, áudio, vídeo (que podem ser trabalhados como diversas fotos em cada segundo de vídeo), dados de telemetria, entre outros.

Um cuidado, independentemente dos dados serem estruturados ou não, que deve ser levado em consideração é a sua limpeza. Se dentro da sua análise a fonte de dados contiver mais dados que são conhecidos como ruídos ou outliers, do que dados que são possíveis de se aproveitar, o andamento da análise será prejudicado e seu resultado poderá sofrer interferências que fogem do controle.

Para facilitar o entendimento, pense somente em dados estruturados. Quando é necessário utilizar uma massa de dados com muitas variáveis

(atributos), e não é de domínio do analista todos os valores e correlações que existem nas variáveis, como devem ser escolhidas as variáveis que de fato farão parte do experimento e então utilizada na criação e validação do modelo? Fazer esta escolha não é uma tarefa simples e cada escolha depende diretamente do problema tratado.

Escolher as variáveis de uma análise necessita que o analista verifique os principais pontos exigidos para responder às questões que foram passadas e foque todos os esforços nas respostas, deixando passar como ruído os aspectos que não estão diretamente ligados à resposta que precisa ser encontrada.

Traçando um paralelo à venda de carros, ao se analisar a potência de um motor e quantos quilômetros é possível percorrer com 1 litro de combustível, a cor do carro ou se ele possui vidros elétricos não influenciam. Pois são variáveis que não estão diretamente ligadas à resposta que está procurando. Já se a necessidade fosse entender quais carros são mais vendidos que outros, aí a cor e se possui vidros elétricos tem correlação. Repare que em ambas perguntas é possível usar a mesma massa de dados para encontrar a resposta, porém cada uma tem um enfoque nas variáveis importantes e descarta os ruídos para aquela análise.

Outra característica pertinente à limpeza de dados é a quantidade de valores inexistentes em uma variável importante para a análise. Depois de escolher quais variáveis farão parte do modelo, é preciso garantir que estas variáveis possuam dados que serão aproveitados. Buscar estes dados não significa que o analista vai deixar na massa de dados somente dados favoráveis à resposta que precisa ser encontrada. Nós somos cientistas de dados, e os cientistas não procuram por respostas que comprovam a hipótese ou a vontade de alguém. Nós cientistas buscamos a resposta através dos dados, seja ela favorável ou não. Os dados não mentem e é nossa obrigação entregar a resposta verdadeira.

Garantir que os dados existem, é verificar qual o percentual de valores inexistentes, os famosos NULL, naquela variável. Um bom número para descartar a variável (característica) da amostra é quando esta característica possui mais de 40% de valores indefinidos – ou ausentes.

## Obtenção de dados

Dados complementares podem ser adquiridos através de *marketplaces*. Empresas como Microsoft e Amazon trabalham com estas lojas de dados, fornecendo um canal único e centralizado para que empresas possam fornecer dados que geram, sejam eles pagos ou gratuitos. Obter estes dados de fontes externas pode auxiliar seu trabalho, tendo o poder de cruzar dados que não são de domínio do seu negócio, mas que tem uma dependência na sua análise.

Voltando ao exemplo de venda de carros, o valor do cambio do dólar americano minuto-a-minuto pode ser interessante para a análise, pois o dólar influencia a importação e peças e também o preço do combustível. E como você é uma montadora de carros, você não tem dados pertinentes ao dólar em sua base, então é viável obter estes dados de algum fornecedor que possa auxiliar neste passo para a análise.

Outro exemplo. Independente do seu negócio é importante conhecer seu público. Seu negócio é virutal e você não possui o endereço de correspondência dos clientes que navegam no seu site. Imagine um portal de notícias que entrega todas as informações gratuitamente e não exige nenhum tipo de cadastro para acessar o conteúdo. Você não tem o endereço de nenhum visitante ao seu portal, mas você tem o IP de cada visita. Existem fornecedores que vendem o endereço de conexão baseado no IP, com isso, é possível descobrir qual a cidade e bairro de cada visitante que esteve na sua página, e assim, direcionar ações específicas direcionadas de acordo com interesses regionais

Se você está começando na área e não tem bases de dados para trabalhar, ou não pode fazer compra de dados através de *marketplaces*, uma saída é procurar em mecanismos de buscas por universidades que fornecem diversas bases de dados gratuitamente para estudo. É possível baixar as bases de dados para aprender uma técnica de ciência de dados (*data science*), aplicando o algoritmo à esta massa de dados e analisando o resultado. Uma das universidades mais procuradas para se obter bases de dados para estudos

é a UCI – *University of California, Irvine*<sup>9</sup>. Que através de seu portal de aprendizado de máquinas e sistemas inteligentes, fornece uma grande quantidade de bases de dados dos mais diversos tipos de finalidades.

Em diversos cenários o volume de dados é bastante extenso e isso pode dificultar a pesquisa e a criação dos modelos. Isso pode acontecer por causa da dificuldade de obtenção dos dados novamente para validar o experimento, ou o volume de dados utilizado é muito grande inviabilizando sua coleta em tempo hábil. Muitas vezes, ao ver uma pesquisa de intenção de votos nos períodos pré-eleição, deixando uma dúvida sobre a veracidade de que aquilo está tendencioso para o lado A ou o lado B. Isso acontece porque a pesquisa é feita com uma amostragem da população.

A estatística é uma área que auxilia os cientistas de dados neste ponto, pois foi provada em diversos experimentos<sup>10</sup> que é possível inferir o resultado da população utilizando somente uma amostragem. Isso realmente funciona quando alguns cuidados são tomados e os analistas entregam o resultado real e não o que agrada quem contratou a pesquisa. Veja alguns pontos que devem ser seguidos em uma pesquisa de intensão e votos para possibilitar que esta amostra represente o universo:

**Estudos multicêntricos:** Em geral é realizado por diversos grupos de pesquisas que trabalham em conjunto, cada um coletando dados da sua região seguindo uma mesma parametrização para coleta dos dados. Isso melhora a generalização dos resultados. O problema enfrentado nesta técnica é que nem sempre os responsáveis pela pesquisa controlam as metodologias aplicadas em todos os centros de pesquisa que estão ajudando naquela coleta dos dados.

**Estudos randomizados:** Deve escolher aleatoriamente os indivíduos que fazem parte da pesquisa. Não se pode escolher indivíduos baseados em

---

9 Página da UCI - <http://archive.ics.uci.edu/ml/>

10 Palestra de Fabiane Nardon no TDC SP 2016 - <http://pt.slideshare.net/tdc-globalcode/tdc2016sp-trilha-data-science-64259824>

uma região de maior influência de pessoas favoráveis à A ou a B, isso torna a pesquisa tendenciosa. A randomização permite resolver esse problema, que é conhecido como viés de auto-seleção. A randomização pode ser feita de forma simples, estratificada ou por conglomerados. A randomização simples coleta dados aleatórios do universo inteiro analisado, podendo enviesar o resultado em determinados momentos (quando somente uma amostra do universo possui determinada característica e esta característica está sendo buscada na análise). A randomização estratificada é aplicada em momentos no qual indivíduos de um mesmo grupo são agrupados por uma variável segregadora e então alguns indivíduos destes grupos são sorteados aleatoriamente. Por fim, a randomização por conglomerados agrupa os indivíduos por alguma variável mais superficial e menos segregadora, e por fim, indivíduos destes conglomerados são sorteados aleatoriamente.

Estas são apenas duas formas que possibilitariam a pesquisa apresentar dados mais próximos à realidade do país, que neste caso, é o universo da amostra.

## Lendo dados externos ao Azure Machine Learning

O *Azure Machine Learning* (AzureML) possibilita trabalhar com diversas origens de dados, desde de um arquivo que contém seus campos separados por vírgulas, até consultas diretas em ambientes do Hadoop utilizando código em Hive. Também é possível criar manualmente sua própria massa de dados ou enviar um lote de arquivos comprimidos. Para trabalhar com os dados externos, encontre os módulos que estão dentro do grupo *Data Input and Output*, no menu da esquerda, como apresentado na Figura 29 a seguir.

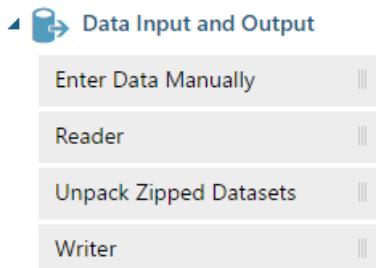


Figura 29 – Módulos de entrada e saída de dados

O processo de leitura de dados externos é realizado através do componente Reader, que deve ser arrastado para dentro do experimento e configurado para acessar a fonte externa. Uma das formas de leitura é através de um endereço direto na URL, onde basta informar o endereço e o processo faz a leitura daquela origem de dados. Voltando às bases de dados públicas da UCI, um exemplo que possui o CSV através da URL é o *Poker Hand Data Set*<sup>11</sup> onde um dos *datasets*<sup>12</sup> contém 1 milhão de observações e 11 variáveis (atributos).

Ao arrastar o módulo *Reader* para o experimento, algumas opções de configuração são apresentadas no painel, à direita da tela (conforme mostra a Figura 30). Estas opções possibilitam que o componente seja configurado de acordo com cada origem desejada.

---

11 *Poker Hand Data Set* -  
<http://archive.ics.uci.edu/ml/datasets/Poker+Hand>  
12 <http://archive.ics.uci.edu/ml/machine-learning-databases/poker/poker-hand-training-true.data>

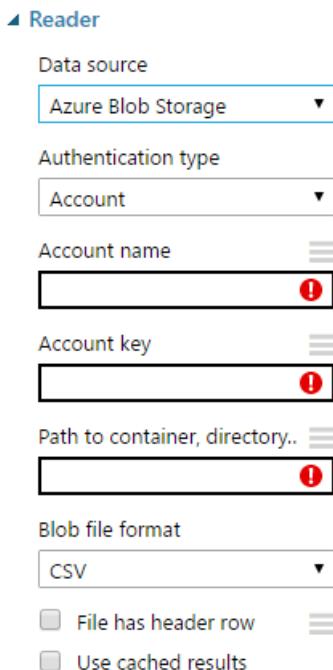


Figura 30 – Propriedades do módulo Reader

Por padrão a opção apresentada é Azure Blob Store, mas a primeira opção que aparece na caixa de seleção da fonte de dados ao ser expandida é de Web URL via HTTP (conforme observado na Figura 31). É com ela que a URL do dataset que possui as mãos do pôquer é informado, e então o componente fará a leitura dos dados deste endereço.

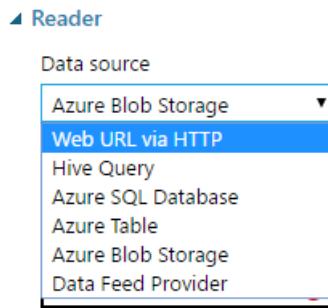


Figura 31 – Opções de fontes de dados disponíveis

Ao informar esta opção de origem dos dados, as opções disponíveis para configuração do módulo se adaptam, e então é necessário preencher alguns campos (conforme demonstrado na Figura 32).

This screenshot shows two side-by-side configurations of the 'Reader' module. Both configurations have 'Web URL via HTTP' selected in the 'Data source' dropdown. The left configuration has an empty 'URL' field with a red exclamation mark icon indicating an error. The right configuration has a correctly filled 'URL' field with the value 'http://archive.ics.uci.edu/ml/'. Both configurations also show 'CSV' selected in the 'Data format' dropdown and two unchecked checkboxes: 'CSV or TSV has header...' and 'Use cached results'.

Figura 32 – Preenchimento dos valores das propriedades para o módulo Reader

Ao informar a URL no campo proposto, e mantendo o formato de dados como um CSV, basta executar o experimento clicando no botão RUN e aguardar o término do processamento. Ao concluir a leitura, um *check* verde é colocado ao lado do módulo, conforme pode ser observado na Figura 33 a seguir.

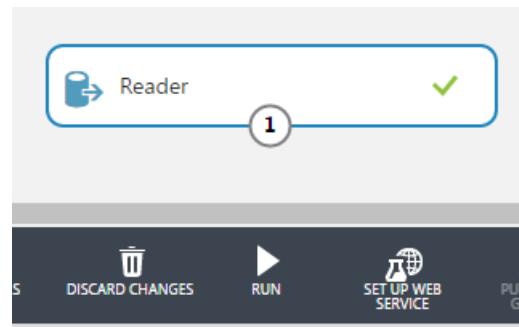


Figura 33 – Módulo Reader executado com sucesso

Ao verificar o resultado de saída do módulo Reader, é possível visualizar o *dataset* que foi lido.

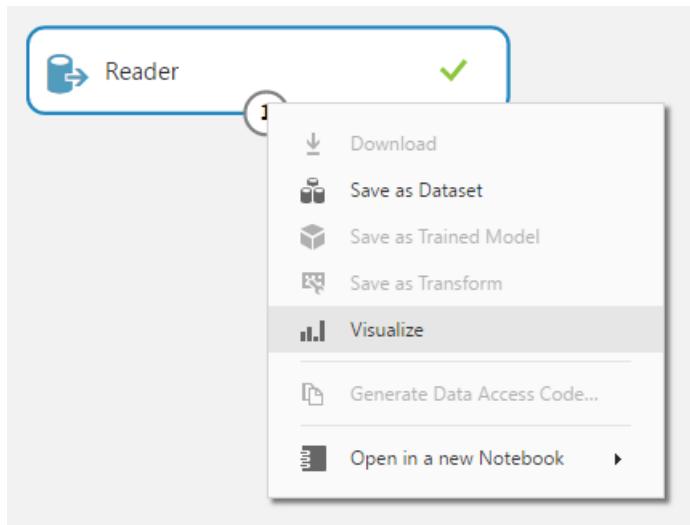


Figura 34 – Visualização dos resultados (dataset lido)

Esta visualização é obtida ao clicar na saída do Reader (círculo com o número 1 na parte de baixo do módulo), conforme visto na figura 34, e então apontar o mouse para a opção *Visualize* no menu de contexto. O resultado é semelhante com este da Figura 35 a seguir.

## Análise preditiva com Azure Machine Learning e R

rows	columns										
1000000	11										
view as	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11
1	1	1	1	13	2	4	2	3	1	12	0
3	12	3	2	3	11	4	5	2	5	1	
1	9	4	6	1	4	3	2	3	9	1	
1	4	3	13	2	13	2	1	3	6	1	
3	10	2	7	1	2	2	11	4	9	0	

Figura 35 – Exemplo de dados lidos com o módulo Reader.

As outras possibilidades de leitura, como: *Hive Query*, *Azure SQL Database*, *Azure Table*, *Azure Blob Storage*, *Data Feed Provider*, também podem ser utilizadas e cada uma possui sua peculiaridade de configuração (parâmetros específicos de cada fonte).

No caso da leitura de dados armazenado em um *Azure Blob Storage*, é preciso ter a chave de acesso ao *storage* para que a conexão seja feita com sucesso. Esta chave é obtida dentro do portal do *Microsoft Azure* nas configurações da *storage account*. A Figura 36 mostra um exemplo de onde obter as chaves de acesso ao *blob*.

## Gerenciar Chaves de Acesso

Quando você gerar novamente as chaves de acesso de armazenamento, precisará atualizar as máquinas virtuais, os serviços de mídia ou os aplicativos que acessam essa conta de armazenamento para usar as novas chaves. [Learn more](#).

NOME DA CONTA DE ARMAZENAMENTO

CHAVE DE ACESSO PRIMÁRIA

**regenerar**

CHAVE DE ACESSO SECUNDÁRIA

**regenerar**

Figura 36 – Chaves de acesso ao Azure Blob Storage

A chave deve ser colocada na configuração do módulo, no respectivo parâmetro. O nome da conta do *storage* deve ser colocada na propriedade *Account Name* do módulo Reader. O valor existente na Chave de Acesso Primária deve ser colocado no campo *Account Key* do módulo Reader, e o caminho que você possui seus dados deve ser informado no campo *Path to Container, directory...* também do módulo em questão. Ao preencher os campos deste módulo completamente, você observará (Figura 37) uma aparência similar à figura que se segue.

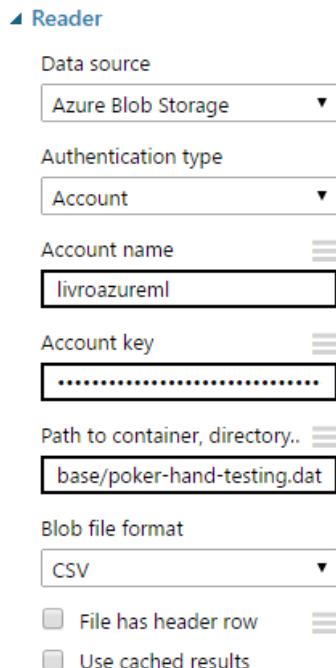


Figura 37 – Propriedades do módulo Reader preenchidas

Ao terminar a configuração você deve seguir para o próximo passo executando o experimento clicando no botão RUN. É esperado que o *check* verde fique preenchido no componente. Então é possível visualizar os dados que foram lidos da origem, seguindo os mesmos passos feito alguns parágrafos acima, na leitura de dados a partir de uma URL.

## Escrevendo dados a partir do Azure Machine Learning

O processo para escrever o resultado a partir de algum trabalho no experimento é tão simples quanto foi ler os dados, a diferença é meramente vetorial – brincadeira com as palavras e seus sentidos, uma vez que ao zerar um vetor ele deve ser percorrido exatamente como feito para seu preenchimento, porém no caminho inverso.

Voltando ao experimento em que foram lidos os dados das mãos de pôquer da origem através de uma URL, é possível escrever o resultado diretamente após a leitura. Porém, para colocar um passo intermediário e mostrar que não foi um simples download da URL para o destino, o componente que fará a nomenclatura das colunas será aplicado e então os dados serão escritos contendo os nomes de colunas ao invés de apenas Col1, Col2, Col3 e assim por diante. Pela documentação fornecida no site da UCI, as 10 primeiras colunas representam as cinco cartas na mão e seus respectivos naipes e a décima primeira coluna é o resultado da sua mão do pôquer naquela jogada.

Para renomear as colunas, é possível utilizar o módulo *Metadata Editor*, que permite trabalhar com alguns elementos relacionadas às colunas. Ele é encontrado no menu da esquerda dentro da categoria de *Data Transformation* (conforme mostrado na Figura 38).

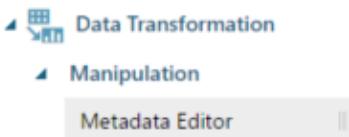


Figura 38 – Menu com a localização do módulo *Metadata Editor*

Ao arrastar este componente para a tela, e ligar a saída do módulo *Reader*, é preciso configurar o comportamento do *Metadata Editor* para que ele reflita a ordem desejada (Figura 39).



Figura 39 – Conectando dois módulos em um experimento

Ao clicar no módulo *Metadata Editor*, é preciso configurar duas propriedades. A primeira é relativa a quais colunas que devem se manter no experimento em sua saída (o círculo na parte de baixo do componente). Então, para informar isso, é necessário clicar no botão *Launch Column Selector* nas propriedades do componente (Figura 40).

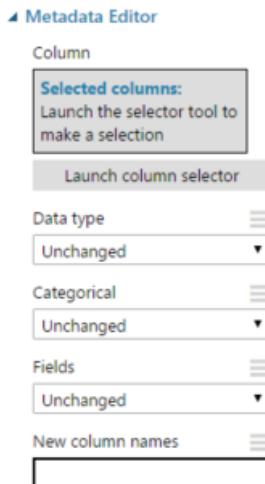


Figura 40 – Propriedades do módulo *Metadata Editor*

Garanta que todas as colunas estão selecionadas na caixa de *available columns* (colunas disponíveis) e então clique no botão > para enviar todas para a caixa de *selected columns* (colunas selecionadas). Também é possível configurar com a opção *with rules*. Em seguida confirme clicando no botão *check* no canto inferior direito da tela, conforme mostrado na Figura 41.

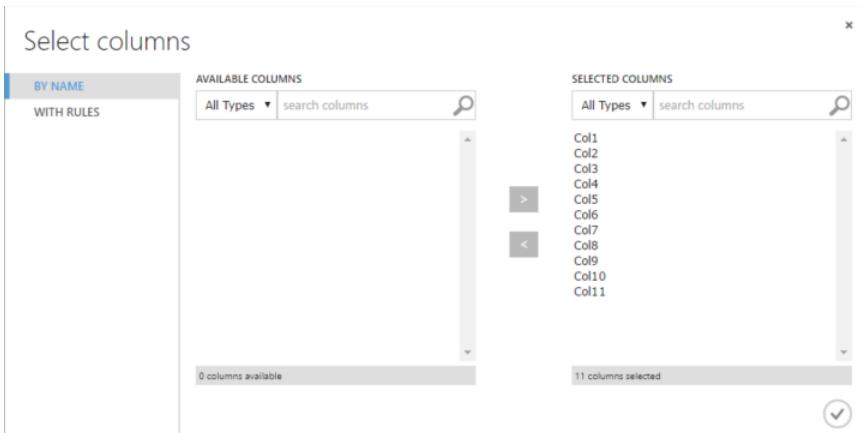


Figura 41 – Seleção de colunas

Em seguida, para renomear cada uma das colunas, é preciso informar os nomes na propriedade *New Column Names*, separando por vírgula na ordem que eles devem substituir os nomes das colunas. Uma possibilidade é colocar o seguinte valor para esta propriedade, e desta forma ser parelho ao que a documentação do conjunto de dados nos informou.

```
Naipe1,Carta1,Naipe2,Carta2,Naipe3,Carta3,Naipe4,Carta4,Naipe5,Carta5,MaoDoPoker
```

Ao término da configuração, as propriedades do módulo *Metadata Editor* será semelhante à mostrada na Figura 42 a seguir.

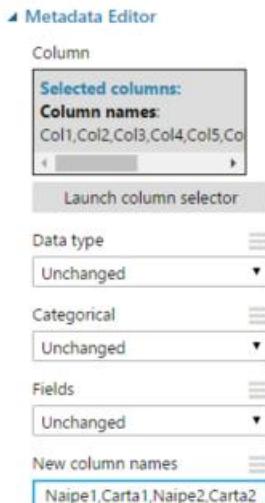


Figura 42 – Propriedades do módulo *Metadata Editor* preenchidos

Ao executar o experimento clicando no botão verde, é esperado que os dois componentes sejam marcados com o *check* verde. Garanta que os nomes das colunas foram alterados clicando no círculo da saída do módulo *Metadata Editor* e em seguida apontando o mouse para *Visualize*. O resultado deve ser parecido com o que é mostrado na Figura 43.

rows	columns
25010	11
view as	
	Naipe1 Carta1 Naipe2 Carta2 Naipe3 Carta3 Naipe4 Carta4 Naipe5 Carta5 MaoDoPoker
1	10 1 11 1 13 1 12 1 1 9
2	11 2 13 2 10 2 12 2 1 9
3	12 3 11 3 13 3 10 3 1 9
4	10 4 11 4 1 4 13 4 12 9
4	1 4 13 4 12 4 11 4 10 9
1	2 1 4 1 5 1 3 1 6 8
1	9 1 12 1 10 1 11 1 13 8
2	1 2 2 2 3 2 4 2 5 8

Figura 43 – Saída do módulo *Metadata Editor*

Para escrever toda esta massa de dados com as colunas renomeadas, é necessário utilizar o módulo *Writer*, que fica próximo ao módulo *Reader* no menu. Arraste o módulo *Writer* para o experimento e então faça a ligação entre a saída do *Metadata Editor* e a entrada do *Writer*. O seu experimento deve parecer com o mostrado na Figura 44.



Figura 44 – Experimento contendo os módulos de leitura e escrita

Neste momento é preciso fazer a configuração do módulo, informando onde os dados serão escritos. Algumas opções são possíveis, como *Hive Query*, *Azure SQL Database*, *Azure Tables* e *Azure Blob Storage*. Seguindo o mesmo processo que foi feito para ler os dados do *Azure Blob Storage*, vamos escrever estes dados no *blob storage*, porém agora com o nome das colunas alterados. A configuração é bastante similar, a diferença é o nome do arquivo que agora será diferente para não substituir o arquivo

original (já existente), que foi colocado lá no exemplo em questão. Garanta que a opção *overwrite* esteja selecionada na propriedade *Azure blob storage write mode*. E também verifique que a opção *Write blob header row* esteja marcada (Figura 43).

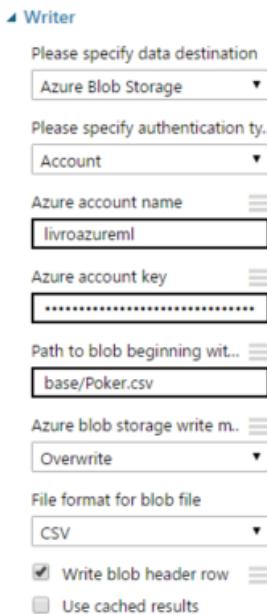


Figura 45 – Propriedades do módulo Writer preenchidas

Ao executar o experimento e aguardar seu processamento, um arquivo com o nome *Poker.csv* será escrito dentro do *container* base da *storage account* utilizada. Ao abrir este arquivo CSV, é possível acompanhar como ficaram os dados escritos a partir do experimento (Figura 44).

A	B	C	D	E	F	G	H	I	J	K
Naipe1	Carta1	Naipe2	Carta2	Naipe3	Carta3	Naipe4	Carta4	Naipe5	Carta5	MaoDoPoker
1	10	1	11	1	13	1	12	1	1	9
2	11	2	13	2	10	2	12	2	1	9
3	12	3	11	3	13	3	10	3	1	9
4	10	4	11	4	1	4	13	4	12	9
4	1	4	13	4	12	4	11	4	10	9
1	2	1	4	1	5	1	3	1	6	8
1	9	1	12	1	10	1	11	1	13	8
2	1	2	2	2	3	2	4	2	5	8
3	5	3	6	3	9	3	7	3	8	8
4	1	4	4	4	2	4	3	4	5	8
1	1	2	1	3	9	1	5	2	3	1
2	6	2	1	4	13	2	4	4	9	0
1	10	4	6	1	2	1	1	3	8	0
2	13	2	1	4	4	1	5	2	11	0
3	8	4	12	3	9	4	2	3	2	1
1	3	4	7	1	5	2	4	4	13	0

Figura 46 – Dados escritos a partir do Azure Machine Learning em um blob no Azure

A escrita pode ser feita em diversos momentos do experimento. Sempre que precisar salvar os dados que estão processados até aquele momento. Outra ideia é salvar o resultado final do experimento, para isso você pode utilizar o módulo *Writer* e ter uma cópia dos dados salvos no destino escolhido.

## 4 – Criando o Primeiro Modelo

### Identificando o problema

Para criar um modelo preditivo que atenda a sua necessidade é importante saber quais problemas devem ser resolvidos com a solução. Após identificar o problema, é preciso escolher qual algoritmo usar para criar esse modelo. Os algoritmos serão tratados com detalhes no Capítulo 07. Por enquanto, imagine que é necessário resolver um problema de classificação binária, ou seja, que possui apenas duas possibilidades de resultado. O problema a ser resolvido ajudará o analista de crédito se deve-se ou não conceder empréstimos para um solicitante baseado em dados do passado de outros solicitantes e seus pagamentos. Este é apenas um exemplo para aplicar as técnicas e facilitar o entendimento. Lembrando que o seu problema real pode ser outro.

Os problemas podem surgir das mais variadas necessidades e em diversas áreas. Em um projeto de alto impacto desenvolvido meses atrás a necessidade de uma empresa era, com base em informação nutricional de alimentos, classificar qual era o grupo que aquele alimento se enquadrava. Isso possibilitou a empresa abrir filiais em outros países mesmo sem conhecer o comportamento alimentar daquele lugar. Usando apenas modelos preditivos foi possível identificar os grupos dos alimentos a partir de sua informação nutricional.

Neste capítulo será trabalhado o exemplo da concessão de crédito a partir de uma base de dados pública da UCI – Universidade da Califórnia-Irvine<sup>13</sup> sobre créditos cedidos a solicitantes na Alemanha alguns anos atrás.

O Dataset possui 1000 observações e 21 variáveis, em outras palavras, são 1000 linhas com 21 colunas cada. As variáveis representam informações relevantes dos solicitantes. Alguns exemplos de dados encontrados nas

---

<sup>13</sup> Base de Dados de Crédito Alemão, na Universidade da Califórnia-Irvine: <http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29>

variáveis são: Duração em meses da conta corrente, Histórico de Crédito, Quantidade solicitada, Sexo e estado civil, idade, entre alguns outros. A vigésima primeira variável é um dado importante, deve ser retirado do treino e em seguida, na validação, utilizado novamente. Ele nos permite utilizar seu valor para validar o resultado. Esta é a coluna que identifica se o solicitante foi um bom ou mau pagador. O valor 1 significa um bom pagador e o valor 2 significa um mau pagador. Após criar o modelo e chegar a um resultado satisfatório este modelo passa a receber valores reais, normalmente, de um ambiente de produção e faz a predição.

Com base no problema que descrevemos acima acompanhe como pode ser criado o experimento no Azure Machine Learning que permite essa análise de risco.

## Componentes necessários

O processo de criação pode iniciar como uma receita de bolo, seguindo os passos para produzir um modelo direcionado a uma solução de problema. Com o passar do tempo e com o aumento da experiência na disciplina de *Machine Learning* seus modelos serão cada vez mais complexos e resolverão cada vez problemas mais difíceis. Por hora, começar o modelo seguindo essa receita nos levará a alcançar o objetivo esperado.

O primeiro passo na grande maioria dos cenários é a aquisição dos dados. Neste momento é necessário ter acesso à fonte de dados para poder fazer a leitura e deixar os dados disponíveis para o experimento. No Capítulo 03 é mostrado como fazer esse processo dentro do *Azure Machine Learning*.

Um passo intermediário entre o passo um e dois ainda é referente aos dados de origem. Muitas vezes as variáveis não são nomeadas e é muito mais fácil trabalhar com cenários nos quais se conheçam as variáveis que são utilizadas. Neste passo intermediário é possível limitar quais são as variáveis daquela origem que farão parte do experimento, e também renomear as variáveis a partir deste ponto para que sejam mais fáceis de serem identificadas.

O segundo passo, numa classificação binária, pode ser considerado como o momento de separar os dados de origem em dois grupos, um para fazer o treino do modelo e outro para fazer a validação do modelo. Normalmente neste processo é utilizado um fator de 70%|30% ou de 80%|20%. Estes fatores são os percentuais relativos à quantidade de dados da origem que será utilizado para treinar o modelo 70% ou 80%, e o restante (30% ou 20%) é pra validar se o modelo está com um resultado favorável. No Capítulo 05 é explicado como funciona essa validação do modelo.

O terceiro passo é uma junção de dois outros processos, nos quais um dos processos é a alimentação do dado separado para o treino e o segundo é o algoritmo que irá processar os dados para criar o modelo. Este processo é popularmente conhecido como Treinar a Base.

O passo seguinte é conhecido como Validação do Modelo, e consiste em comparar o resultado obtido com o algoritmo treinado e os dados que foram “retirados” lá no passo dois. Assim, as observações conhecidas que foram trabalhadas no treino saem de cena ficando somente aquelas que não foram utilizadas para criar o modelo preditivo. Então, o algoritmo processa estes novos dados (que são apresentados pela primeira vez ao algoritmo) e entrega o resultado sendo o valor previsto, que é apresentado como *Scored Label* e qual é a probabilidade deste valor estar certo, que é apresentado como *Scored Probabilities*. Estes dois resultados quando analisados juntos com o valor da variável conhecida pelo retorno podem lhe dar uma direção se o algoritmo atende a sua necessidade ou não.

Um quinto passo pode existir. Não é sempre que ele é utilizado nos experimentos, mas como este cenário é para aprender o processo, é possível comparar o resultado de dois algoritmos diferentes. Isso ajuda a escolher qual é o melhor algoritmo para este cenário. No Capítulo 07 é apresentada uma tabela contendo um guia para ajudar a direcionar a escolha do algoritmo.

Por fim, ao escolher o melhor algoritmo para este experimento deve-se salvar o modelo treinado e utilizar este modelo para receber os valores reais do sistema. Este recebimento dos valores pode ser feito por processo em lote – batch – ou então em tempo real – real time – dependendo da necessidade

de aplicação. Em tempo real é possível consumir uma API que o próprio Azure Machine Learning fornece.

Agora que já foram apresentados os elementos essenciais para criar o modelo é hora de colocar em prática este assunto.

## Criando o primeiro modelo

Ao abrir o *AzureML Studio* crie um experimento novo. Pode chamar de **Análise de Risco**, como na Figura 47.

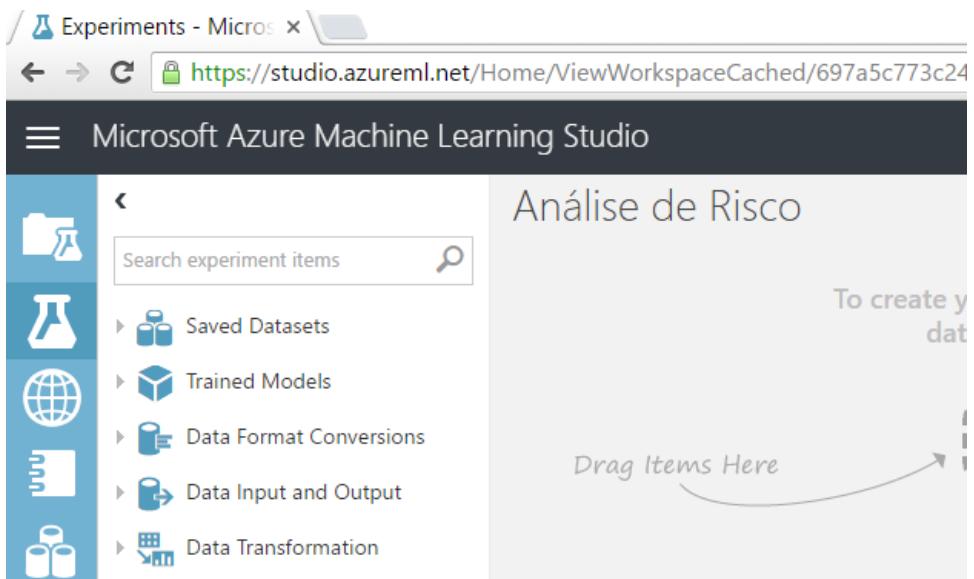


Figura 47 - Criando o experimento Análise de Risco

Seguindo os passos comentados no subcapítulo anterior é necessário informar onde está a origem dos dados. Você pode importar os dados para seu experimento de algumas formas, como explicado no Capítulo 03 – lendo dados externos. Neste exemplo o Dataset será lido direto a partir dos exemplos existentes no menu *Saved Datasets*, que é uma cópia da URL de

origem<sup>14</sup>. Coloque no seu experimento o dataset **German Credit Card UCI Dataset**, conforme a Figura 48.

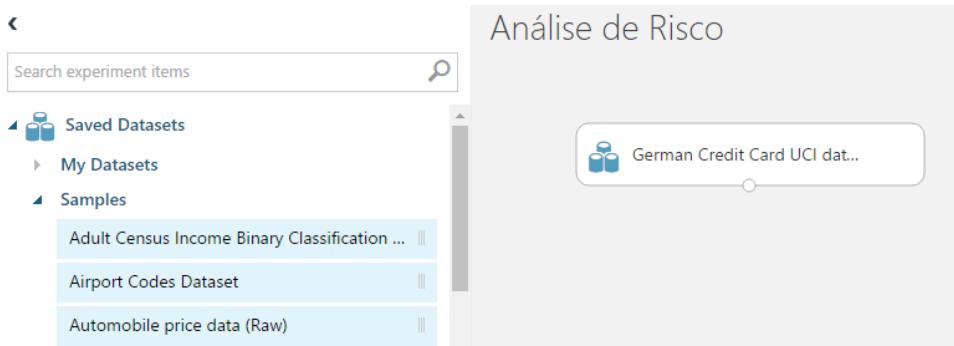


Figura 48 - Dataset do Experimento

Ao visualizar a leitura dos dados o resultado é semelhante ao da Figura 49. Repare na quantidade de observações (linhas - Rows) e de variáveis (colunas - columns).

Análise de Risco > German Credit Card UCI dataset > dataset

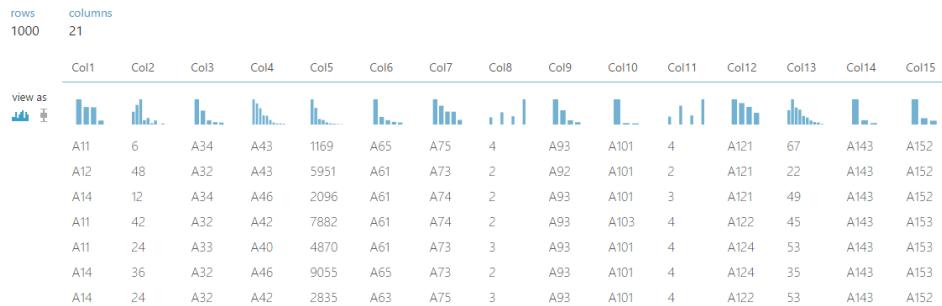


Figura 49 - Visualização dos dados brutos

As colunas não estão nomeadas. É difícil saber o que significa cada um dos valores existentes. Para facilitar este processo é possível renomear as colunas utilizando o componente Edit Metadata. Arraste este componente e

---

<sup>14</sup> URL de Origem dos dados: <http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data>

faça a conexão da saída do dataset para a entrada deste elemento. Garanta que o seu experimento esteja como na Figura 50.

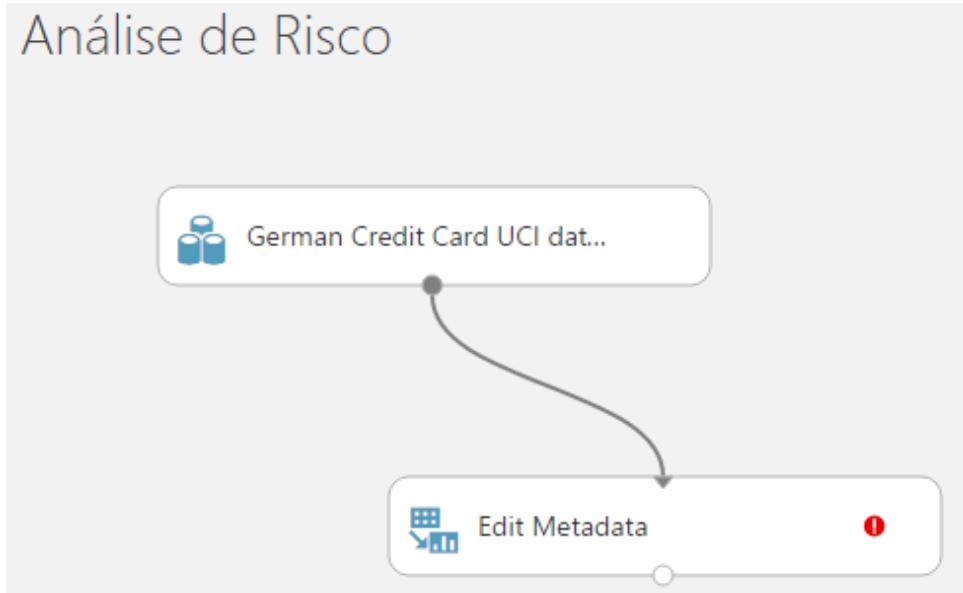


Figura 50 - Conexão do Metadata

Para alterar os nomes das colunas é necessário informar quais serão os novos nomes de cada coluna separados por vírgula. As colunas deste dataset são as seguintes:

- Status da Conta Corente
- Duração em Meses
- Histórico de Crédito
- Propósito do Crédito
- Total de Crédito
- Título de Capitalização
- Funcionário Desde

- Percentual do Rendimento Disponível
- Sexo e Status Civil
- Outros Débitos ou Garantias
- Residente desde
- Propriedade
- Idade
- Outros Parcelamentos
- Habitação
- Número de créditos neste banco
- Empregado
- Responsável por quantas pessoas
- Telefone
- Trabalhador Estrangeiro
- Risco ao Crédito (variável preditiva)

Estas definições de variáveis podem ser encontradas na documentação do dataset, no site da UCI.

Caso precise, volte ao Capítulo 3 – Lendo dados externos para saber como renomear as colunas. Após fazer isso a visualização do dataset será como a Figura 51.

## Análise preditiva com Azure Machine Learning e R

Análise de Risco > Edit Metadata > Results dataset

rows  
1000    columns  
21

Status Conta Corrente	Duracao em Meses	Historico de Credito	Proposito	Total de Credito	Titulo de Capitalizacao	Funcionario Desde	Percentual do Rendimento Disponivel	Status Civil e Sexual
A11	6	A34	A43	1169	A65	A75	4	A93
A12	48	A32	A43	5951	A61	A73	2	A92
A14	12	A34	A46	2096	A61	A74	2	A93
A11	42	A32	A42	7882	A61	A74	2	A93
A11	24	A33	A40	4870	A61	A73	3	A93
A14	36	A32	A46	9055	A65	A73	2	A93
A14	24	A32	A42	2835	A63	A75	3	A93
A12	36	A32	A41	6948	A61	A73	2	A93
A14	12	A32	A43	3059	A64	A74	2	A91

Figura 51 - Dataset com as colunas renomeadas

Após ter o dataset renomeado, que faz parte ainda daquele passo intermediário entre o passo um e dois, é o momento de executar o segundo passo: aplicar o fator de separação dos dados para obter dois grupos. Um será usado para treinar o modelo e o outro para validar o treino. Para isso, arraste o componente Split Data para dentro do seu experimento, conecte a saída do componente Edit Metadata na entrada do Split Data e informe o fator de separação para 0.8 conforme apresentado na Figura 52.

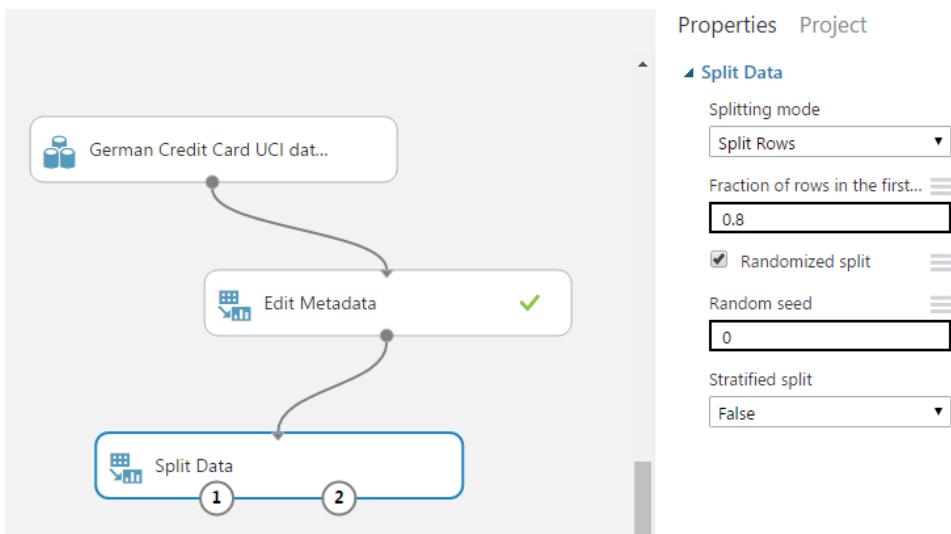


Figura 52 - Fator de separação dos dados

A partir de agora, a saída 1 do componente Split Data possui 80% de dados que entraram no componente e a saída 2 possui os outros 20% que sobraram. O processo para treinar o modelo é composto de duas etapas, sendo uma o recebimento dos dados para treinar (a saída 1 do componente Split Data) e a outra o algoritmo que será utilizado. Arraste o componente Train Model para o seu experimento e faça uma conexão da saída 1 do Split Data na entrada número 2 do componente. Para a entrada 1 é necessário escolher o algoritmo. Existem diversos algoritmos prontos que podem ser utilizados. Vamos utilizar o Two-Class Support Vector Machine, que é explicado em detalhes no Capítulo 07 – Algoritmos. Arraste este componente para o experimento e conecte a saída dele na entrada 1 do modelo *Train Model*, conforme a Figura 53.

## Análise de Risco

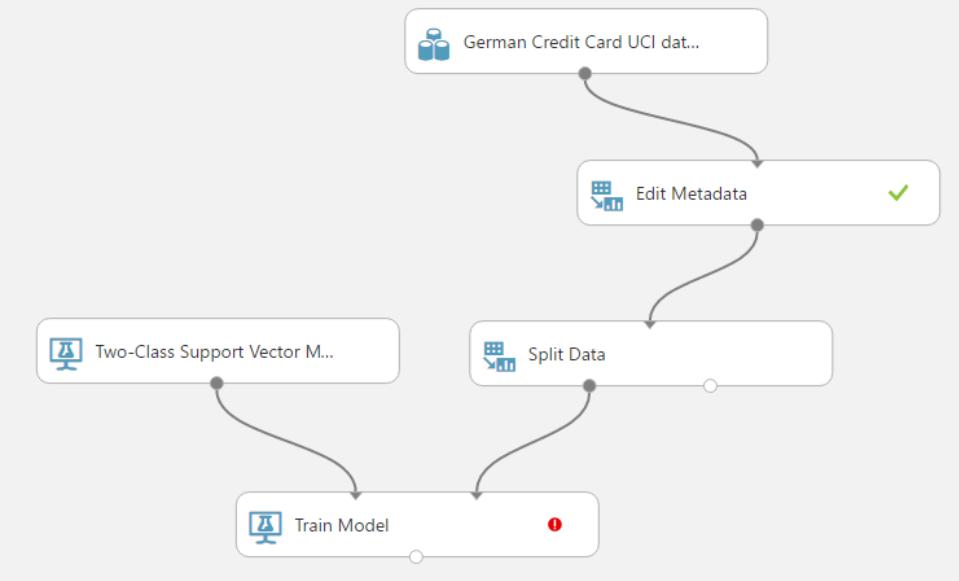


Figura 53 - Treinando o modelo com um SVM

O componente Train Model precisa de uma configuração para informar ao experimento qual é a coluna que será utilizada para previsão. Voltando um pouco no livro, nas colunas (variáveis) que o dataset possui, a última coluna é a Risco ao Crédito. É esta coluna que deve ser informada ao componente para que ele faça a previsão. Acompanhe nas Figuras 54 e 55 como é este processo.

## ▲ Train Model

Label column

**Selected columns:**

Launch the selector tool to make a selection

Launch column selector

Figura 54 - Selecionar as colunas

Select a single column



Figura 55 - Coluna de Risco ao Crédito selecionada

Seguindo os passos da receita de bolo do subcapítulo anterior, é necessário validar se o algoritmo classifica com sucesso ou falha a predição. Para isso, um novo componente é inserido no experimento. Arraste para o experimento o componente de Score Model. Conecte a saída do modelo treinado na primeira entrada do novo componente. Lembra dos 20% de dados que sobraram do Split Data? Ele será utilizado agora conectando a segunda saída do Split Data na segunda entrada do Score Model. Ao executar é possível visualizar a saída do Score Model, e encontrar os valores de Score Label e Score Probabilities nas últimas colunas apresentadas. Acompanhe a Figura 56.

## Análise preditiva com Azure Machine Learning e R

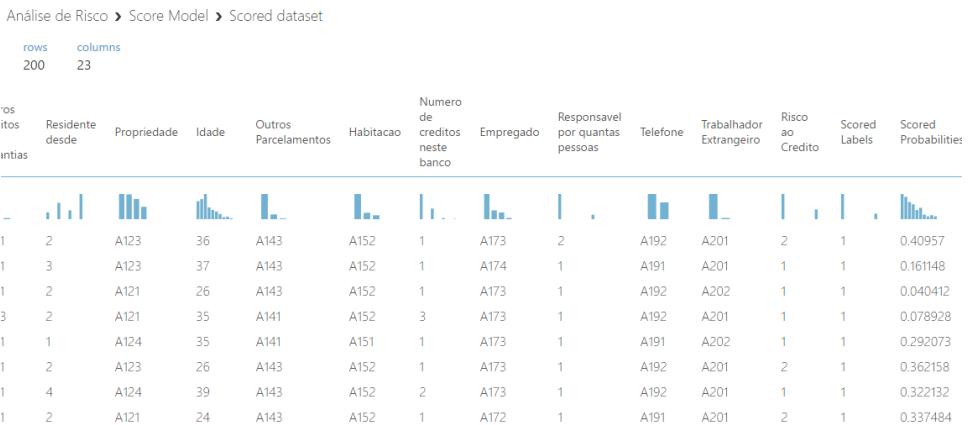


Figura 56 - Resultado do Label e Probability

O quinto passo, que existe em alguns experimentos, é a avaliação do modelo que está explicada em detalhes no Capítulo 05. Para aplicar esta avaliação, arraste o componente Evaluate Model para o experimento e conecte a saída do Score Model na primeira entrada do componente Evaluate Model. Conforme a Figura 57.

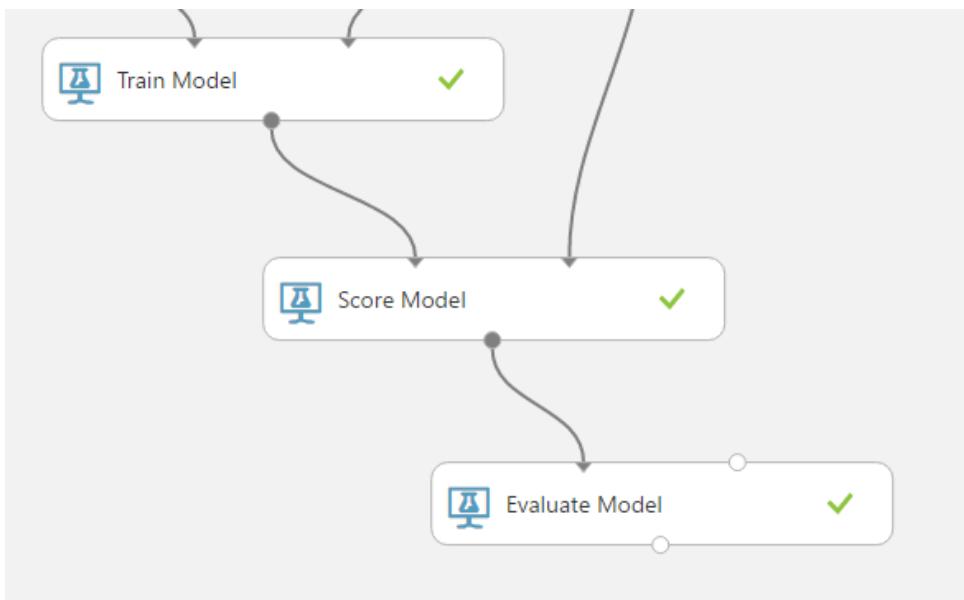
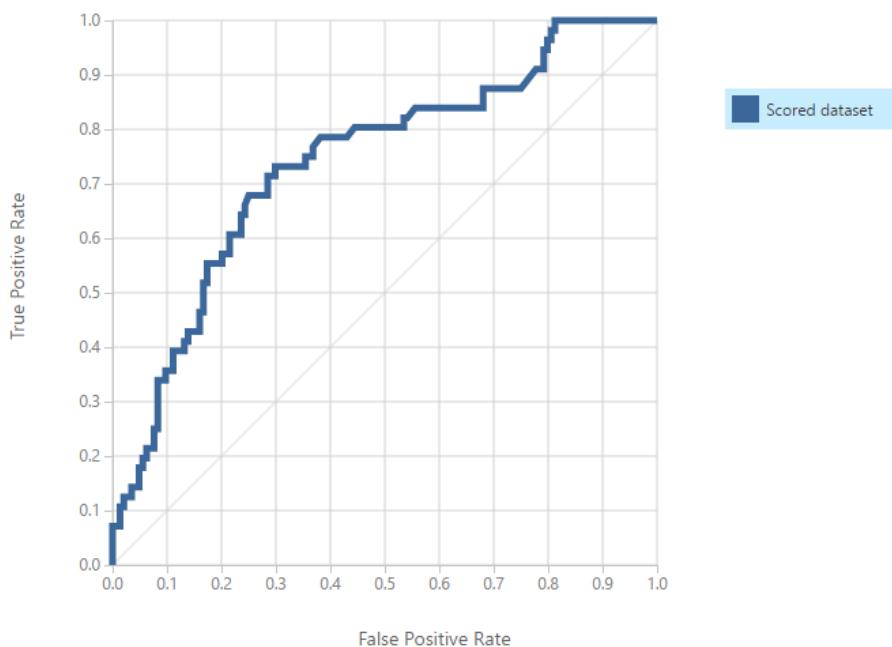


Figura 57 - Validando o modelo

Ao visualizar o resultado, é possível acompanhar as curvas ROC, Precision/Recal e Lift conforme a Figura 58. Lembrando que elas estão detalhadas no Capítulo 5 – Validando seu modelo.

Análise de Risco > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
20	36	0.740	0.556	0.5	0.740
False Positive	True Negative	Recall	F1 Score		
16	128	0.357	0.435		

Figura 58 - Curva ROC

Reparam que o componente Evaluate Model permite mais de uma entrada, isso possibilita comparar treinos realizados com algoritmos diferentes. Para fazer esta comparação é necessário voltar alguns passos e adicionar outros componentes ao experimento. Devemos adicionar um novo Train Model, um novo Score Model e o novo Algoritmo que será utilizado.

Veja estes componentes arrastados para dentro do experimento, mas sem nenhuma conexão com outros componentes na Figura 59.

Análise de Risco

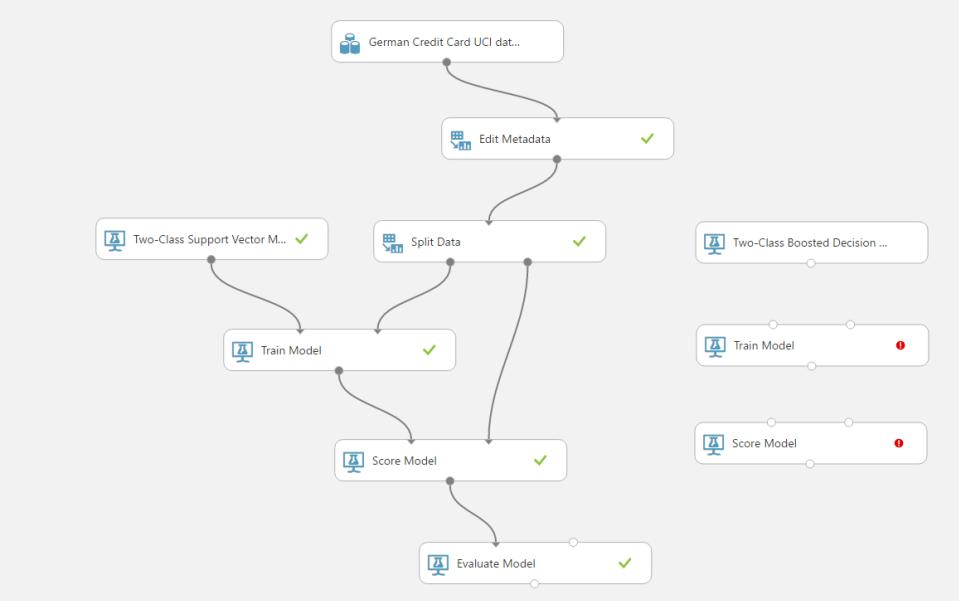


Figura 59 - Novos componentes no experiment

O processo de conexões destes novos componentes seguem o mesmo padrão das conexões que foram feitas anteriormente. Sendo a saída do algoritmo (Two-Class Boosted Decision Tree) conectada na primeira entrada do treino (Train Model). A saída do treino (Train Model) conectada na primeira entrada do componente de validação (Score Model). A saída da validação (Score Model) conectado à segunda entrada do componente de avaliação (Evaluate Model). Acompanhe estas conexões na Figura 60.

### Análise de Risco

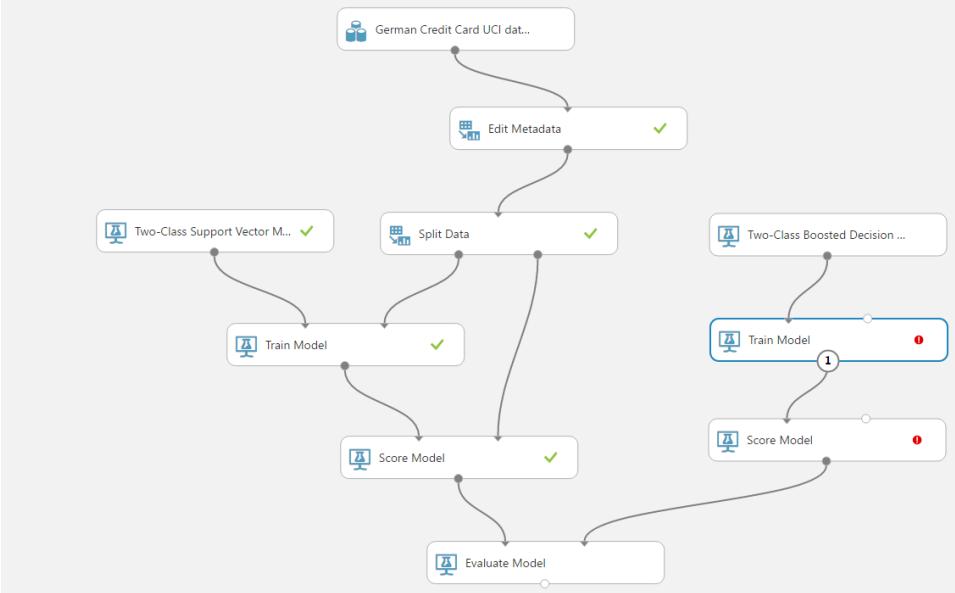


Figura 60 - Novas conexões entre os components

As conexões com os dados também são necessárias e foram deixadas para este momento para facilitar o entendimento de onde estas devem ser conectadas no fluxo de atividades do experimento.

O algoritmo deve utilizar os 80% da base original para criar o modelo. Estes 80% dos dados originais estão na saída 1 do componente Split Data. Arraste um novo conector entre a saída 1 do componente Split Data e a entrada 2 do segundo componente Train Model adicionado a pouco. Veja na Figura 61 como isso deve ficar.

### Análise de Risco

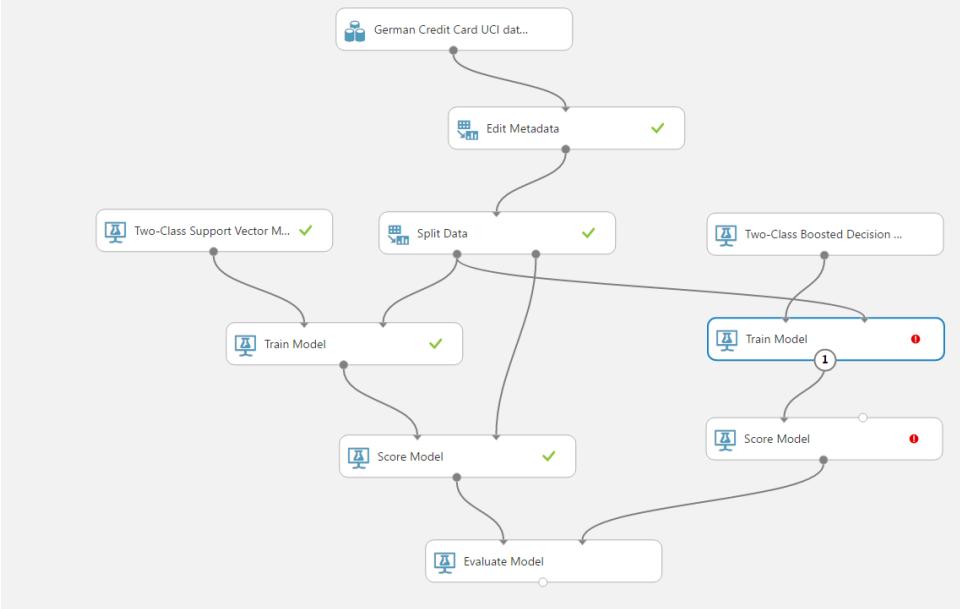


Figura 61 - Conexão com os dados de treino

A validação do algoritmo utilizado para treinar o modelo é feita com aqueles 20% remanescente dos dados originais. Estes dados estão na saída 2 do componente Split Data. Faça uma conexão entre esta saída e a entrada 2 do componente Score Model que faz parte do segundo bloco de treino. Veja na Figura 62 como fica esta conexão.

### Análise de Risco

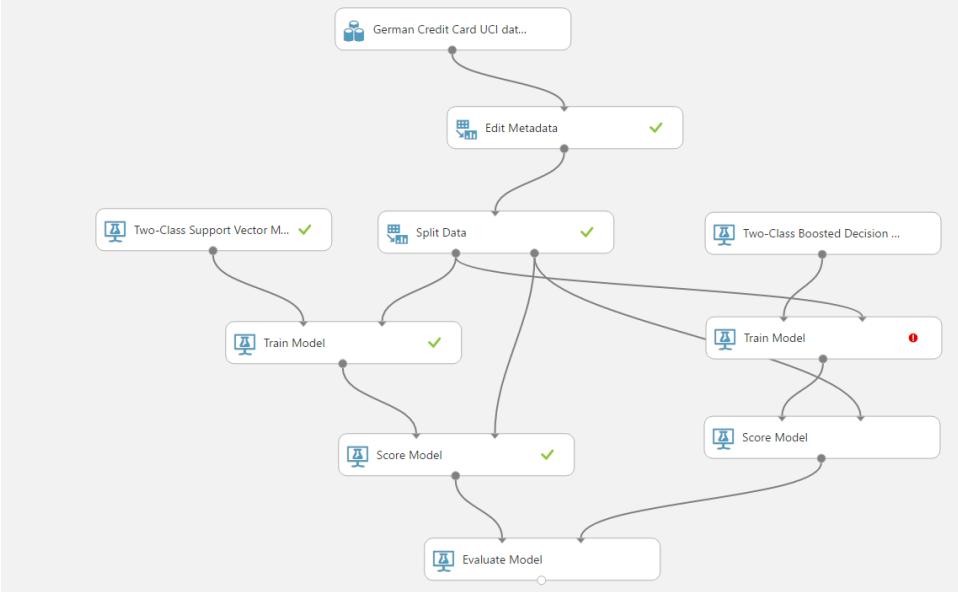


Figura 62 - Todas conexões feitas

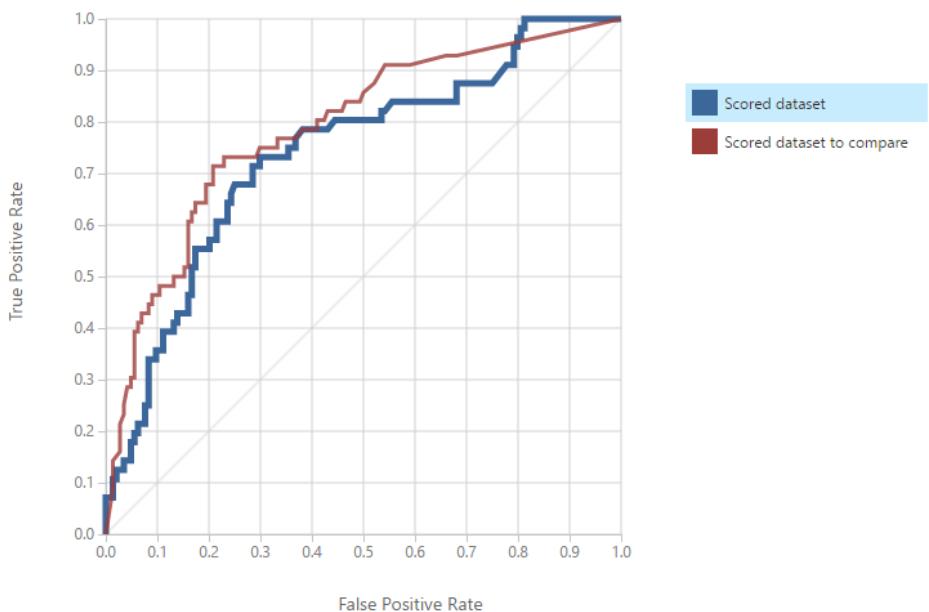
É importante garantir que cada saída do componente Split Data possua duas conexões de destino permitindo que os 80% de dados sejam utilizados para treinar os modelos de ambos algoritmos, e que os 20% de dados de validação também sejam utilizados para os dois blocos distintos.

O segundo componente Train Model precisa ser configurado da mesma forma que foi feito no primeiro componente. Abra as configurações do componente e informe a coluna (variável) Risco ao Crédito. Caso queira lembrar, acompanhe as Figuras 54 e 55 já apresentadas.

Ao executar o experimento e tudo funcionar, os componentes ficarão com uma marca de check verde. Clique na saída do componente Evaluate Model e visualize as curvas de ambos algoritmos utilizados no experimento. Veja nas figuras 17 e 18 a comparação dos resultados obtidos respectivamente com o algoritmo Two-Class Support Vector Machine e Two-Class Boosted Decision Tree.

## Análise de Risco > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT

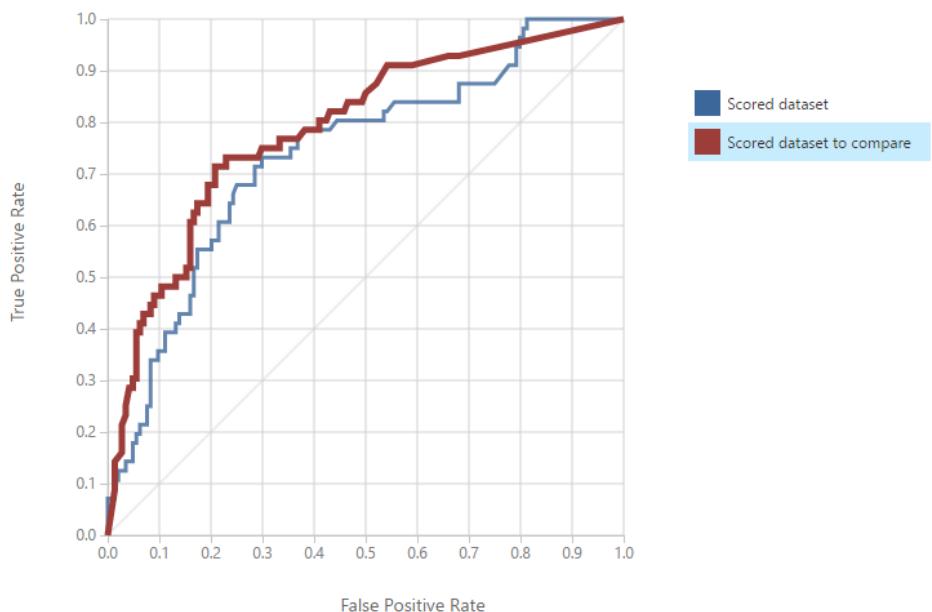


True Positive	20	False Negative	36	Accuracy	0.740	Precision	0.556	Threshold	0.5	AUC	0.740
False Positive	16	True Negative	128	Recall	0.357	F1 Score	0.435				

Figura 63 - Curva ROC do SVM

## Análise de Risco > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT



True Positive	31	False Negative	25	Accuracy	0.760	Precision	0.574	Threshold	0.5	AUC	0.793
False Positive	23	True Negative	121	Recall	0.554	F1 Score	0.564				

Figura 64 - Curva ROC da Árvore de Decisão

Ficou claro, após analisar os dois algoritmos, que o resultado com a Árvore de Decisão foi melhor do que com o Support Vector Machine.

É possível continuar a fazer testes. Caso queira, siga eliminando o algoritmo perdedor (que teve pior resultado) e adicione outro em seu lugar. Avalie o resultado do algoritmo vencedor comparando com o novo, e eleja qual é o vencedor desta batalha. Faça isso com os algoritmos que achar pertinente. No final, você terá um modelo que foi comparado com diversos

algoritmos e que apresenta o melhor resultado até o momento. Existe uma outra forma de melhorar o resultado, que é trabalhando na configuração do algoritmo. Isso é coberto em detalhes no Capítulo 07 – Algoritmos.

O último passo da criação do modelo é a permissão de acessos externos para uso. Vamos seguir com somente estes dois algoritmos testados, dos quais a Árvore de Decisão foi vencedor. Para disponibilizar este modelo para produção é necessário salvar o modelo treinado, e criar um segundo experimento que utilizará este modelo. Vá até a saída do componente Train Model eleito vencedor e aponte para Save as Trained Model (Figura 65), então, uma caixa de diálogo irá surgir pedindo o nome do modelo treinado (Figura 66). Após informar o nome, o seu modelo passa a ficar disponível no menu Trained Models (Figura 67).

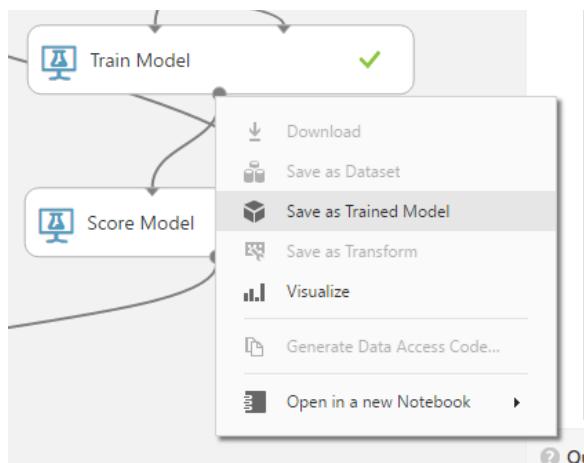


Figura 65 - Salvando o modelo

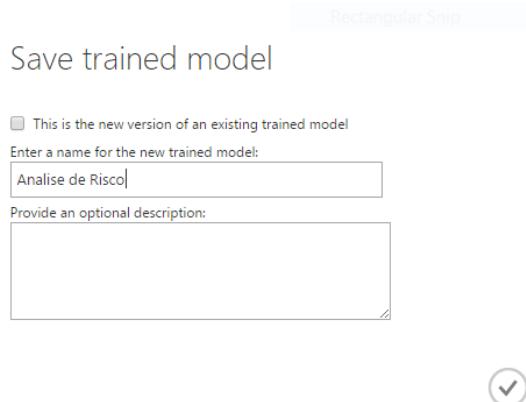


Figura 66 - Nomeando o modelo



Figura 67 - Encontrando o modelo

Para consumir o modelo treinado que foi salvo, um segundo experimento pode ser criado. Ao criar este novo experimento, como o objetivo é receber informação de algum lugar podendo ser um processo em batch ou então em tempo real, e utilizar o modelo preditivo que foi treinado anteriormente, ele terá uma estrutura semelhante ao que foi feito até o momento neste capítulo.

Este novo experimento pode ser chamado de Liberação de Crédito, ou algum nome mais sugestivo que você possa utilizar.

O primeiro componente a ser utilizado é o modelo treinado que foi criado neste capítulo. Se você seguiu a mesma nomenclatura do livro o seu novo experimento terá uma aparência como a Figura 68.

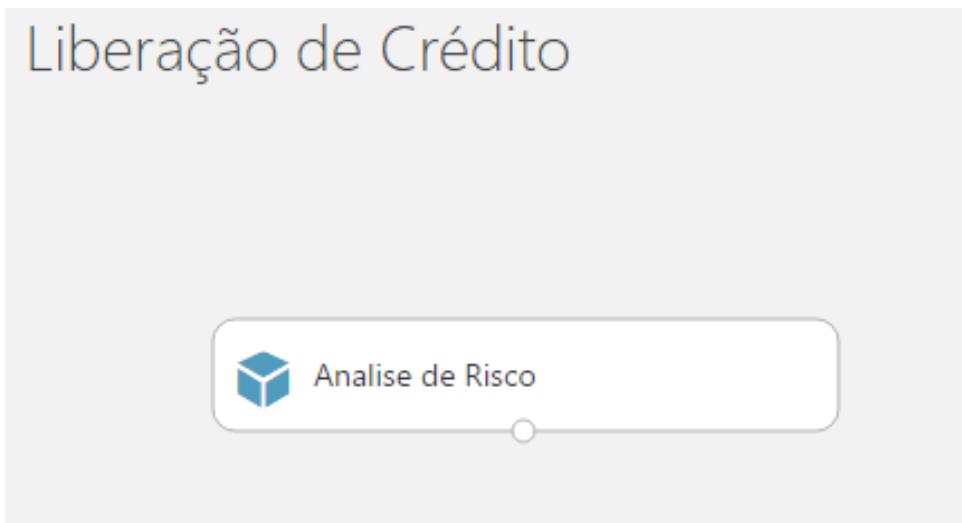


Figura 68 - Novo Experimento

Novamente o componente Score Model será utilizado. Ele deve receber os dados externos que serão disparados contra o modelo treinado na sua entrada número 2, e também o modelo que foi desenvolvido conectado na sua entrada número 1. Para fins de teste e entendimento do processo, novamente, o dataset German Credit Card UCI Dataset será utilizado, mas desta vez com o objetivo de simular um processamento em batch de dados de produção.

É importante que o dataset que será enviado em lote para o modelo treinado também tenha as colunas renomeadas. Faça o mesmo processo realizado anteriormente no modelo *Edit Metadata* para que a saída do dataset original e a entrada no componente Score Model sofra esta manipulação de nomes e você consiga acompanhar o resultado.

Um novo componente deve ser incluído no experimento, é o Select Columns in Dataset. Ele permite que sejam selecionadas as colunas que

estarão na saída deste componente. No nosso dataset de exemplo existe a coluna que tem o valor referente ao cliente indicando se ele é um bom ou mau pagador. Esta coluna é a que foi renomeada para Risco ao Crédito. Arraste este componente para o experimento, conecte a saída do Edit Metadata na entrada dele e abra as configurações. Na seleção de colunas, envie todas as colunas para a saída, com exceção da coluna Risco ao Crédito. Nós não queremos uma influência desta coluna no uso simulado de um ambiente real. Remover essa coluna deixa o dataset que estamos trabalhando agora como um conjunto de dados reais, pois segue a mesma estrutura de dados sem existir a coluna que nós estamos querendo predizer. Veja esta seleção de colunas na Figura 69.

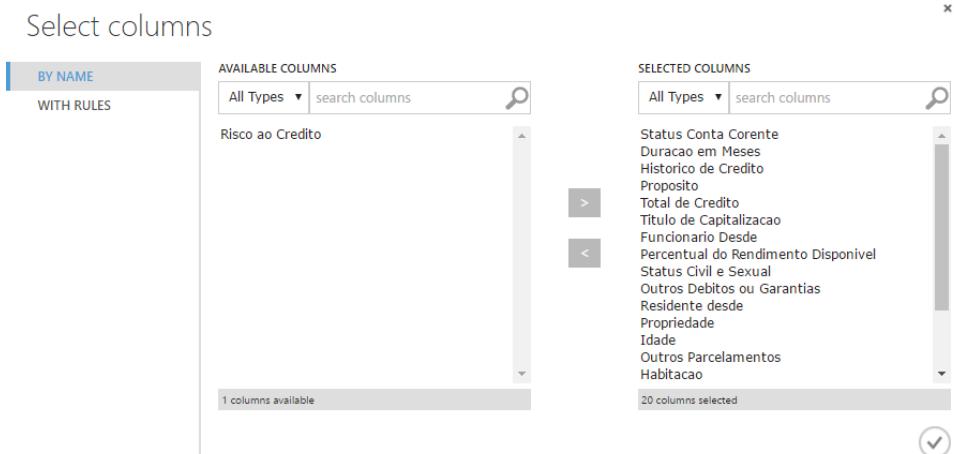


Figura 69 - Seleção de colunas

Ao conectar a saída deste componente ao Score Model, o experimento fica semelhante ao da Figura 70.

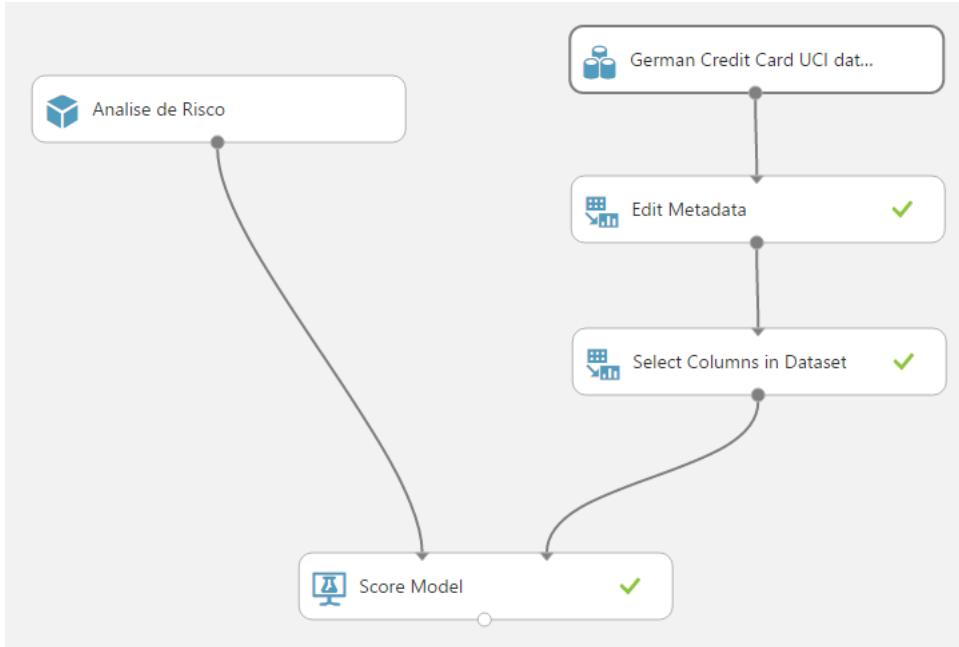


Figura 70 - Experimento com modelo

Por fim, depois de aplicar o modelo preditivo nos novos dados recebidos, é possível escrever a saída em um componente como foi visto no Capítulo 03 – Trabalhando com Dados externos. Para finalizar este processo da criação do primeiro modelo será feita a escrita em um Azure Blob Storage. Confira na Figura 71 a versão final deste experimento de Liberação de Crédito com execução em lote.

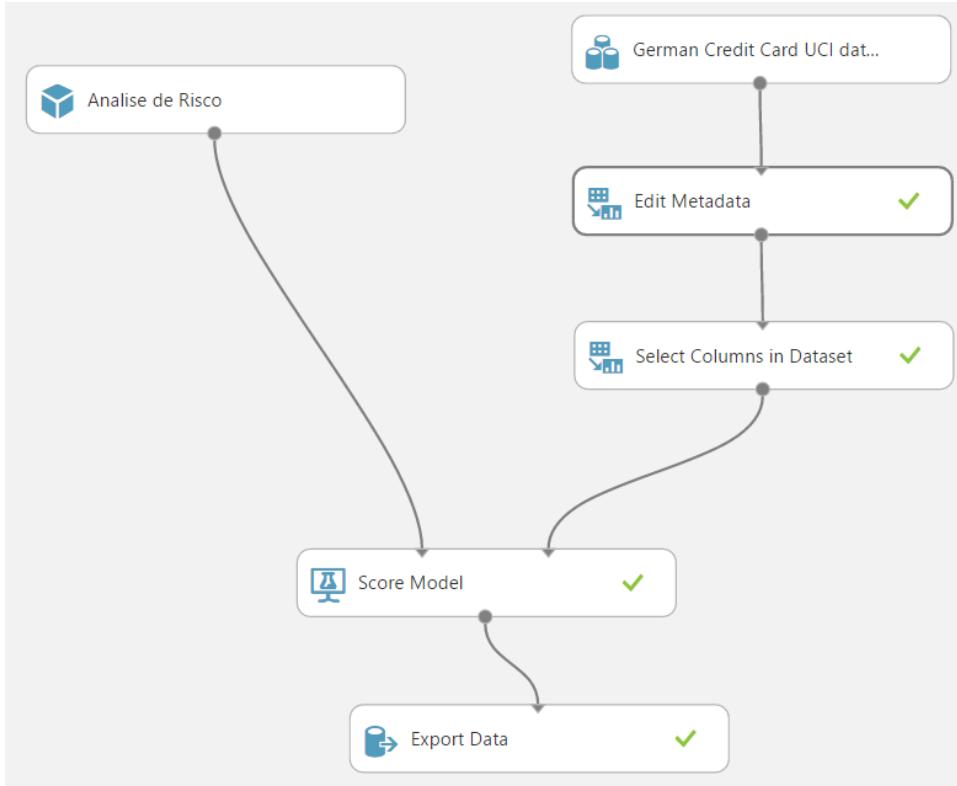


Figura 71 - Escrita do resultado da execução em lote

Uma outra forma de consumir o modelo preditivo é utilizar dados em tempo real. Para não alterar o experimento que faz o processamento em lote clique no botão Save As do experimento criado agora, e renomeie para Liberação de Crédito WS. Veja como é feito isso na Figura 72.



Figura 72 - Duplicando o experimento

Garanta que está utilizando este terceiro experimento, e então clique no botão Setup Web Service que fica localizado na barra inferior do AzureML Studio. Localize-se como na Figura 73.



Figura 73 - Botão de configuração do Webservice

Neste momento, alguns componentes são adicionados automaticamente em seu experimento permitindo que Web Services façam interações tanto com a entrada dos dados quanto com a saída. Acompanhe a Figura 75 onde os componentes foram adicionados.

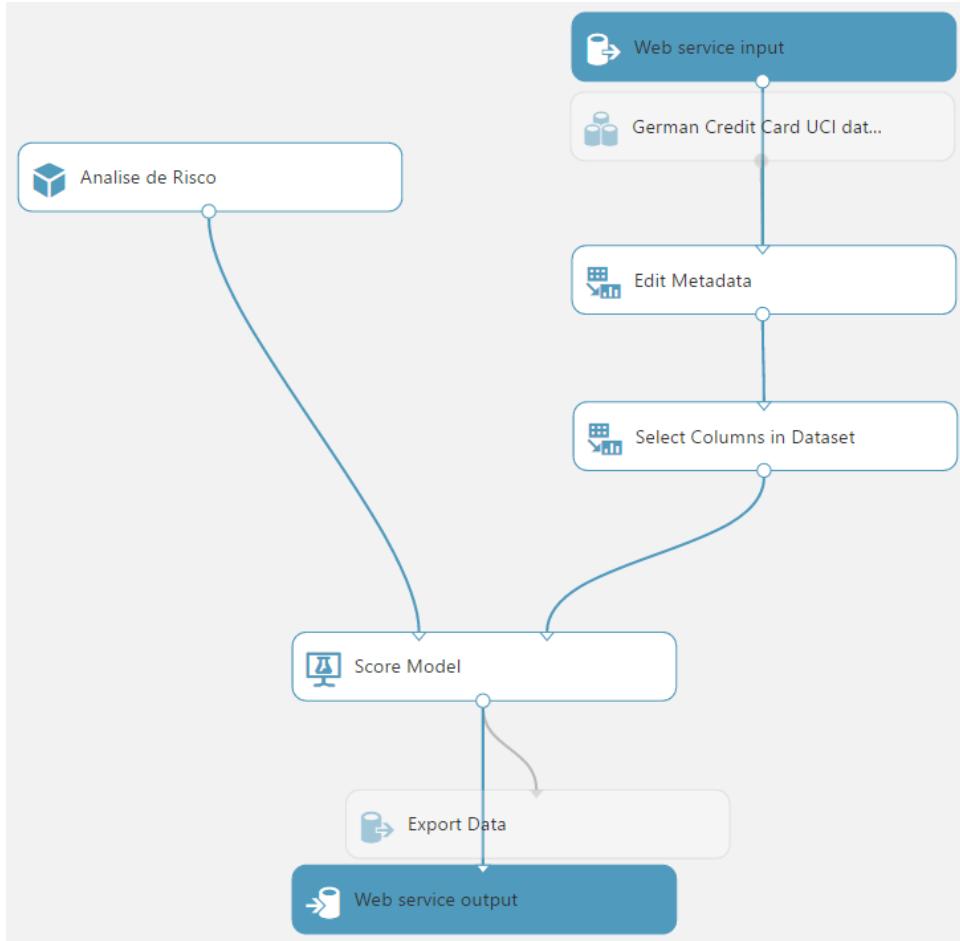


Figura 74 - Componentes do Webservices

Depois de salvar o projeto, repare que onde existia o botão Setup Web Service agora está um botão chamado Deploy Web Service. Veja este botão na Figura 75.

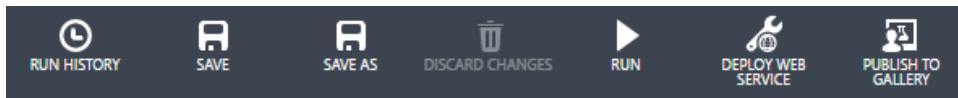


Figura 75 - Deploy Web Service

Ao clicar neste botão de Deploy Web Service, seu experimento será enviado para uma nova área do *Azure Machine Learning* e pode ser acessado diretamente pelo menu lateral que agrupa todos os Web Services existentes no seu Studio. Este menu é o apresentado na Figura 76.

A screenshot of the Azure Machine Learning Studio sidebar. On the left, there is a vertical list of categories: PROJECTS, EXPERIMENTS, WEB SERVICES (which is selected and highlighted in blue), NOTEBOOKS, DATASETS, TRAINED MODELS, and SETTINGS. To the right of the sidebar, under the heading "web services", is a list of deployed web services. The first item in the list is "Liberação de Crédito WS".

NAME
Liberação de Crédito WS
[redacted]

Figura 76 - Web Services disponíveis

Ao clicar no Web Service que foi criado pelo seu experimento, é possível configurar diversos integradores com Endpoints distintos. Inclusive com código pronto que acelera a integração com desenvolvimento de software em C#, Python e R.

## Organizando os Experimentos em Projetos

Três experimentos foram criados, cada um com um objetivo, porém todos são para um mesmo propósito. Identificar bons e maus solicitantes de crédito. Para estes experimentos não ficarem espalhados dentro do Studio é possível organizar criando Projetos e associando os experimentos a esses projetos.

Para criar um projeto clique no ícone de criação de novo objeto, aponte para Projeto e dê um nome. Observe a Figura 77.

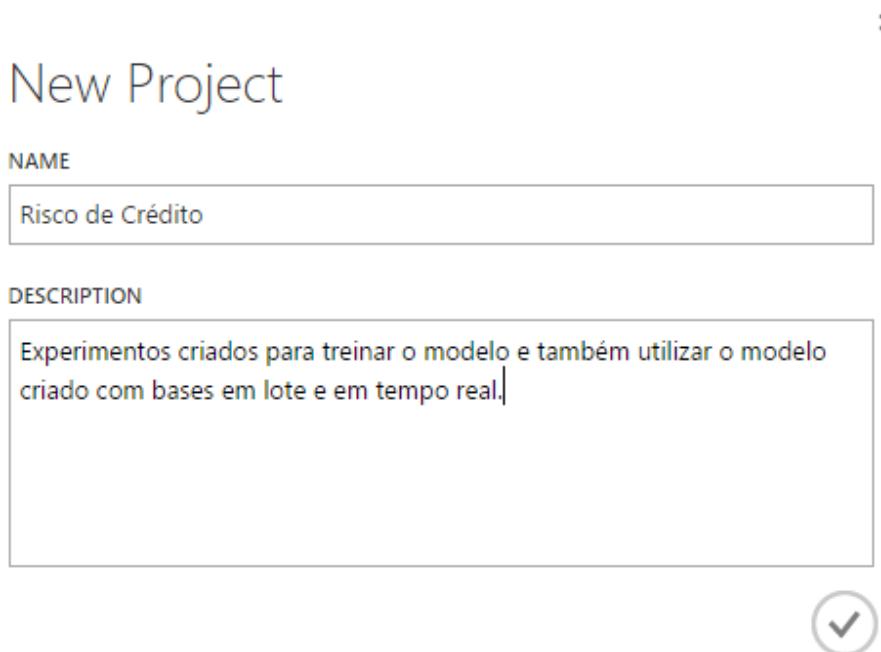


Figura 77 - Novo Projeto

Ao criar o projeto, ele fica acessível através do menu de projetos, conforme a Figura 78.

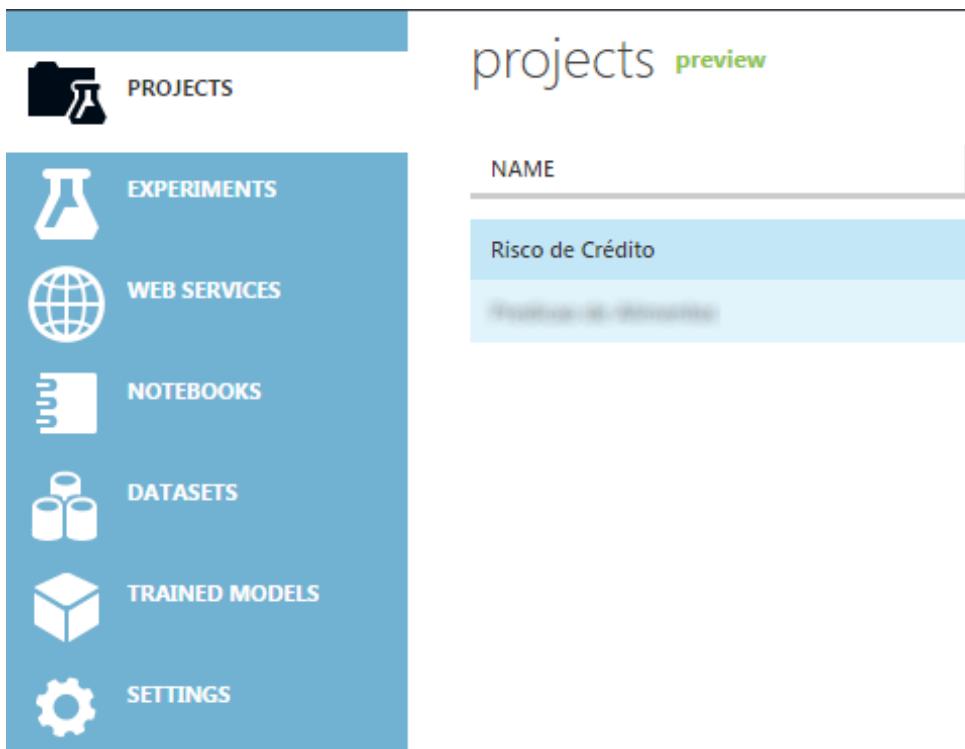


Figura 78 - Projetos disponíveis

Ao abrir o projeto que foi criado, uma tela contendo todos os elementos que fazem parte deste projeto é apresentada. Ao clicar no botão Add Assets uma nova janela é aberta e permite que sejam selecionados quais objetos fazem parte deste projeto. Os objetos criados estão separados dentro de suas categorias, como os Experimentos, os Datasets, Modelos Treinados, Web Services, etc.

Ao selecionar os elementos que fazem parte do projeto e enviar para a caixa ao lado, é possível acompanhar todos os objetos que este projeto engloba. Veja os objetos que criamos neste capítulo na Figura 79.

## Change Project Configuration

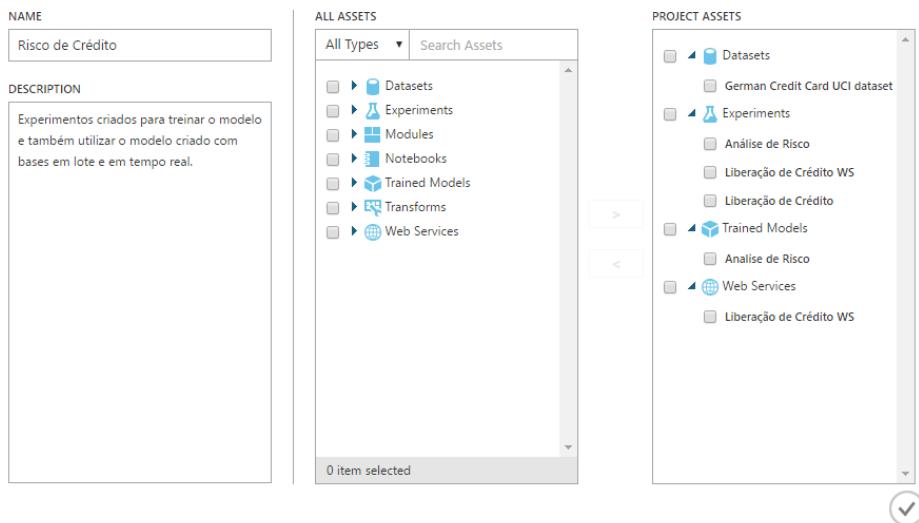


Figura 79 - Objetos referentes à Risco de Crédito

Por fim, na tela de detalhes do projeto todos os elementos que foram selecionados possuem seus detalhes apresentados. A Figura 80 aborda esta tela.

The screenshot shows the Azure Machine Learning Studio interface. On the left is a vertical sidebar with icons for different project components: folder (EXPERIMENTS), flask (WEB SERVICES), globe (DATASETS), document (TRAINED MODELS), and gear (SETTINGS). The main area displays the project details for 'risco de crédito'.

- DESCRIPTION:** Experimentos criados para treinar o modelo e também utilizar o modelo criado com bases em lote e em tempo real.
- EXPERIMENTS:**
  - ▶ Análise de Risco
  - ▶ Liberação de Crédito WS
  - ▶ Liberação de Crédito
- WEB SERVICES:**
  - ▶ Liberação de Crédito WS
- TRAINED MODELS:**
  - ▶ Analise de Risco
- DATASETS:**
  - ▶ German Credit Card UCI dataset

Figura 80 - Detalhes do Projeto

## 5 – Validando um Modelo no AzureML

A construção de um modelo de aprendizagem de máquina pode ser bastante rápido ao se utilizar uma plataforma como o AzureML, mas como podemos ter certeza de que o nosso modelo está no caminho correto e/ou funciona satisfatoriamente?

O primeiro pensamento que devemos ter é que, nos problemas do mundo real, não existe um modelo perfeito que irá acertar a resposta correta em 100% dos casos. Conforme vimos no capítulo do dia a dia do cientista de dados, o importante é encontrar uma solução aceitável para o nosso problema.

Existem diversas técnicas e conceitos utilizados para validar um modelo. No *Azure Machine Learning* podemos usar os módulos *Evaluate Model* e *Cross Validate Model*.

### Evaluate Model

No *Azure Machine Learning* o componente básico que usamos para avaliar o nosso modelo é o *Evaluate Model*. A Figura 81 mostra o módulo sendo usado no *Studio*.

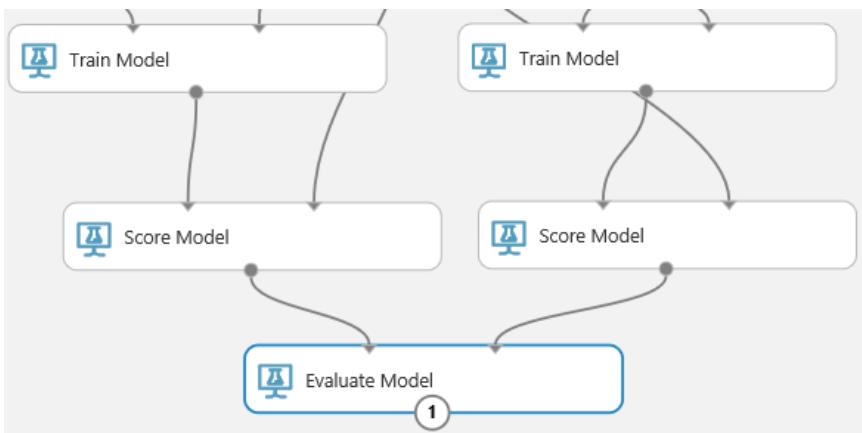


Figura 81 – Usando o módulo Evaluate Model

O *Evaluate Model* pode ser utilizado para avaliar um único modelo, ou realizar a comparação entre dois modelos ao mesmo tempo. A Figura 82 ilustra um exemplo de saída com dois modelos sendo avaliados. Para obter essa visualização basta clicar na saída do componente, representado na Figura 1 com destaque da saída com o número 1, e ir em seguida no item Visualizar, encontrado como *Visualize*.

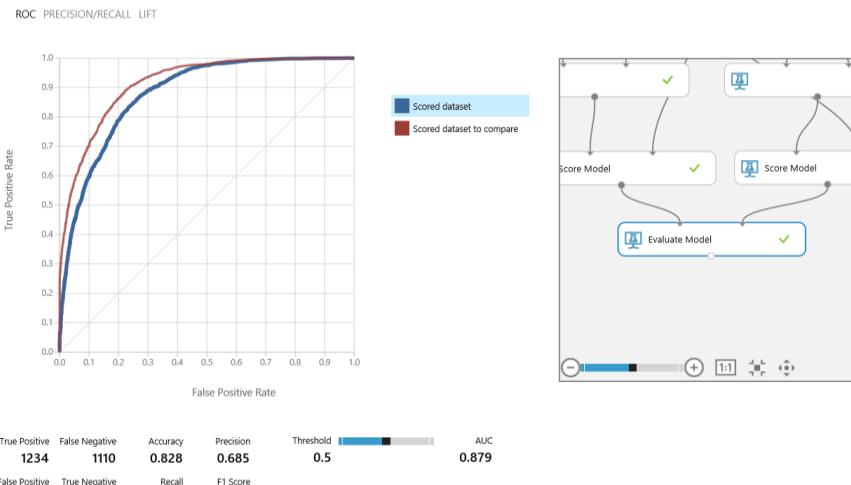


Figura 82 – Exemplo da saída do módulo Evaluate Model

O módulo exibe um *preview* da localização do componente no experimento, bem como diferentes métricas para avaliação do(s) mesmo(s). Nas seções a seguir iremos explorar cada uma destas métricas e como é feito o seu cálculo, bem como conceitos importantes para o melhor entendimento dos seus experimentos.

## Validando Modelos

A seguir temos a explicação de algumas formas de validar, mensurar e comparar modelos.

### Matriz de Confusão

Uma matriz de confusão é uma representação dos testes realizados no formato de matriz. Onde um eixo representa a quantidade das classes classificadas corretamente e o outro as classes encontradas pelos algoritmos utilizados. Problemas de duas classes geram uma matriz 2x2, problemas de três classes geram matrizes de 3x3, e assim por diante. Podemos ter valores absolutos (quantidades) ou relativos (percentuais) nestas matrizes.

Matrizes de confusão funcionam para aprendizagem supervisionada (em geral para problemas de classificação), pois necessitamos conhecer as classes corretas para a comparação com a classe encontrada.

Uma matriz de confusão em um problema de duas classes recebe algumas nomenclaturas especiais. As duas classes são nomeadas entre: classe positiva (também mostrada com o valor 1) e classe negativa (também mostrada com o valor 0).

True Positive    False Negative

1234	1110
------	------

False Positive    True Negative

568	6856
-----	------

Positive Label    Negative Label

>50K	<=50K
------	-------

		PREDICTED LABEL	
		1	0
TRUE LABEL	1	506 TRUE POSITIVE (TP)	112 FALSE NEGATIVE (FN)
	0	169 FALSE POSITIVE (FP)	420 TRUE NEGATIVE (TN)

Figura 83 – Exemplos de duas matrizes de confusão para problemas de duas classes

As posições na matriz também recebem uma nomenclatura especial:

**Verdadeiro Positivo** – VP (*True Positive* – *TP*): Representa a quantidade de elementos da classe positiva classificados corretamente.

**Verdadeiro Negativo** – VN (*True Negative* – *TN*): Representa a quantidade de elementos da classe negativa classificados corretamente.

**Falso Positivo** – FP (*False Positive* – *FP*): Representa a quantidade de elementos negativos que foram classificados como positivos. Ou seja, valores classificados erroneamente.

**Falso Negativo** – FN (*False Negative* – *FN*): Representa a quantidade de elementos positivos que foram classificados como negativos. Ou seja, valores classificados erroneamente.

Uma matriz de confusão em um modelo “perfeito” é aquela onde a diagonal principal da matriz contém todos valores (diferentes de 0), e todas as demais posições contém apenas zeros (sem erro).

A Figura 84 ilustra uma matriz de confusão de um problema de múltiplas classes. A figura foi cortada e levemente adaptada para se encaixar ao formato do livro. Nela podemos dizer que 85,1% dos exemplos que eram da classe “A” foram corretamente classificados como sendo da classe “A”, também podemos dizer que 0,5% dos testes onde era a classe “A” foram classificados como classe “G”, e assim por diante. É um dos mecanismos mais poderosos para verificar visualmente qual as classes que seu modelo não desempenha bem.

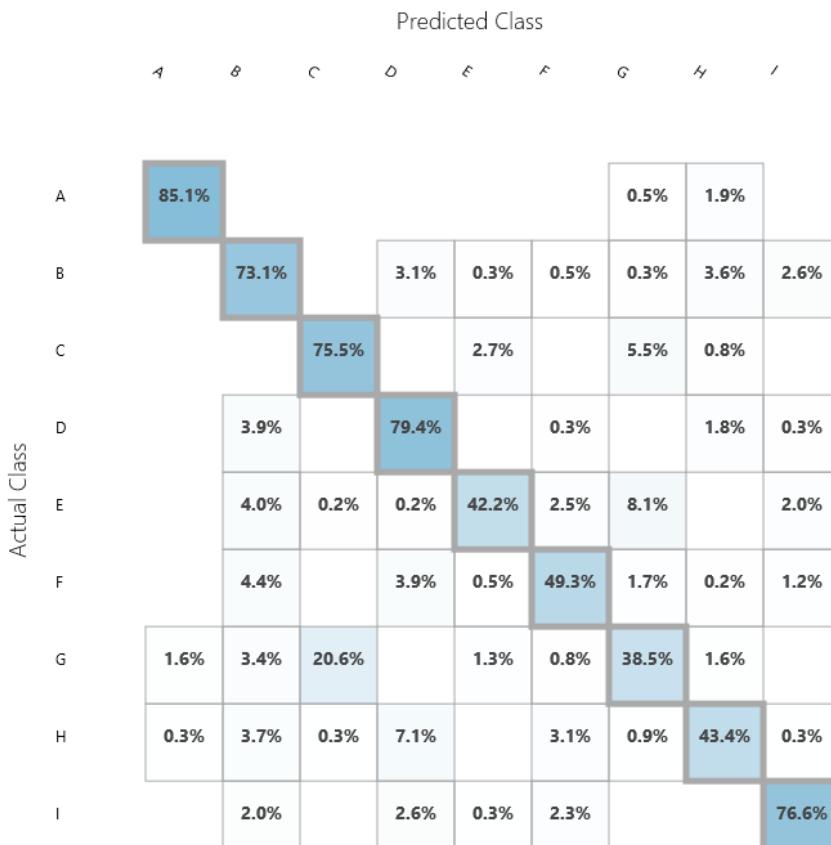


Figura 84 – Exemplo da saída de uma matriz de confusão do módulo Evaluate Model de um problema de múltiplas classes.

## Acurácia

Acurácia, do inglês *Accuracy*, é a razão entre o total de classes retornadas corretamente pelo total de classes classificadas.

$$\text{Acurácia} = [\text{exemplos classificados corretamente}] / [\text{total de experimentos}]$$

Para um problema de duas classes, usando a nomenclatura especial, teríamos a seguinte definição:

$$\text{Acurácia} = VP + VN / (VP + FP + VN + FN)$$

Um modelo “perfeito” seria aquele que a acurácia é de 100% todas as vezes. Em geral apenas problemas muito simples, ou modelos errados, retornam 100% de acurácia.

Nem sempre modelos com maior acurácia são melhores que problemas com menor acurácia, veja mais na seção de precisão a seguir.

## Precisão

Precisão, do inglês *Precision*, retrata a razão entre os valores positivos retornados corretamente (VP) e o somatório dos exemplos classificados como positivos (VP + FP).

$$\text{Precisão} = VP / (VP+FP)$$

Um modelo com uma acurácia menor, mas uma precisão maior pode ser mais adequada, do que um problema com acurácia maior. Logicamente isso dependerá do problema sendo avaliado.

Por exemplo, um teste de paternidade, para dizer que uma pessoa é pai de outra o grau de confiança / precisão precisa ser extremamente elevado. Já um modelo que detecta a existência de uma doença em um exame de rotina pode ter uma precisão um pouco mais baixa que o do teste de paternidade, pois podemos fazer outros exames mais específicos na sequência para confirmação.

## Recall

Recall é o número de classes positivas identificadas corretamente. Podemos utilizar essa métrica para entender o quão completo nossos resultados são.

$$Recall = VP / (VP + FN)$$

### F1-Score

A medida F1 é a média harmônica entre Precisão e Recall. Caracterizada como uma medida mais equilibrada e frequentemente utilizada para comparação de modelos.

$$F1 = 2 * Precisão * Recall / (Precisão + Recall)$$

### Curva ROC e AUC

A curva ROC, sigla de *Receiver Operating Characteristic*, é um mecanismo fundamental para avaliar classificadores binários, que são estes problemas de apenas duas classes.

Em uma curva ROC a taxa de verdadeiro positivo, conhecido como sensibilidade, é plotada em função da taxa de falso positivo, também chamado de especificidade, para diferentes pontos de corte dos parâmetros. O resultado é que quanto mais área coberta na curva ROC, melhor é o modelo.

A área debaixo da curva origina uma medida de mesmo nome conhecida como AUC, sigla de *Area Under the Curve*, e serve para representar de forma numérica para facilitar a comparação. Neste caso, quanto maior a AUC, melhor.

Na Figura 55, temos duas curvas ROC plotadas, podemos perceber que a curva em vermelho possui uma maior área sob a curva, logo podemos afirmar que o modelo em geral é melhor que o representado pela curva em azul.

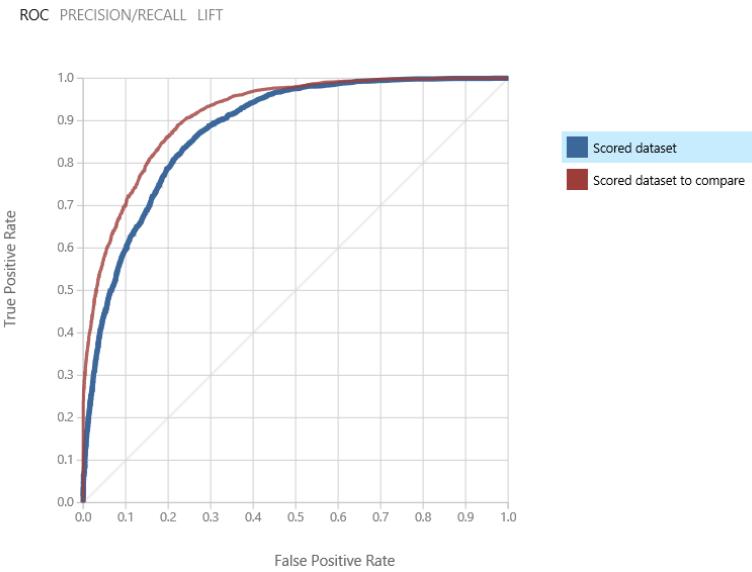


Figura 85 – Curva ROC com resultado de dois modelos

Para referência do que seriam bons valores para a métrica AUC é possível utilizar a seguinte relação. Novamente, isso dependerá do problema abordado.

- 1,0: Predição perfeita
- 0,9: Excelente predição
- 0,8: Boa predição
- 0,7: Predição medíocre
- 0,6: Predição pobre
- 0,5: Predição aleatória
- < 0,5: Existe algum problema grave com o modelo

## Precision/Recall e Lift

Além da curva ROC, o AzureML exibe o gráfico *Precision/Recall* e o gráfico/curva *Lift*.

O gráfico *Precision/Recall* exibe a correlação entre estas duas medidas, conheça este gráfico na Figura 86.

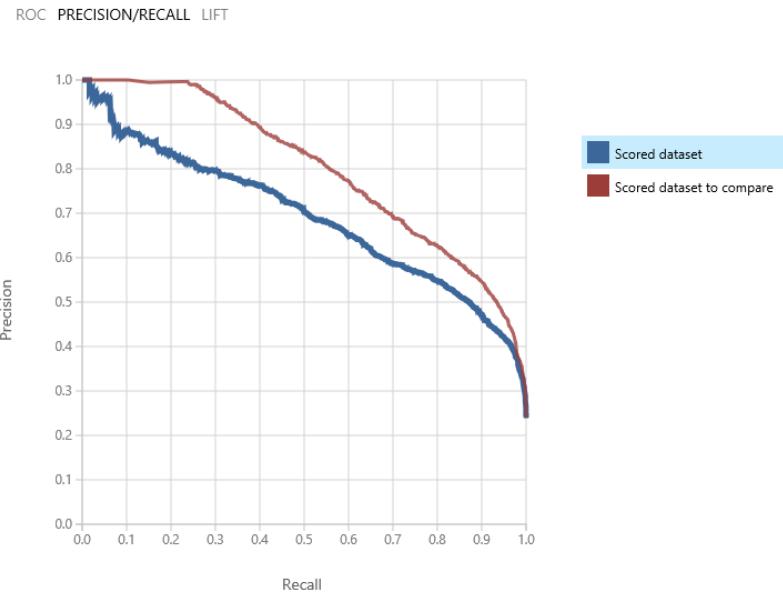


Figura 86 - Correlação entre precisão e recall

O gráfico/curva Lift pode ser considerado uma variação da curva ROC, na econometria pode ser chamada de *Lorenz* ou *Power Curve*. A medida é a indicação de quanto uma predição é melhor/mais correta do que um modelo trivial que classifica aquela classe. Definimos como sendo a taxa de acerto dividida pelo percentual de exemplos da classe positiva. Um exemplo de curva Lift pode ser observado na Figura 87.

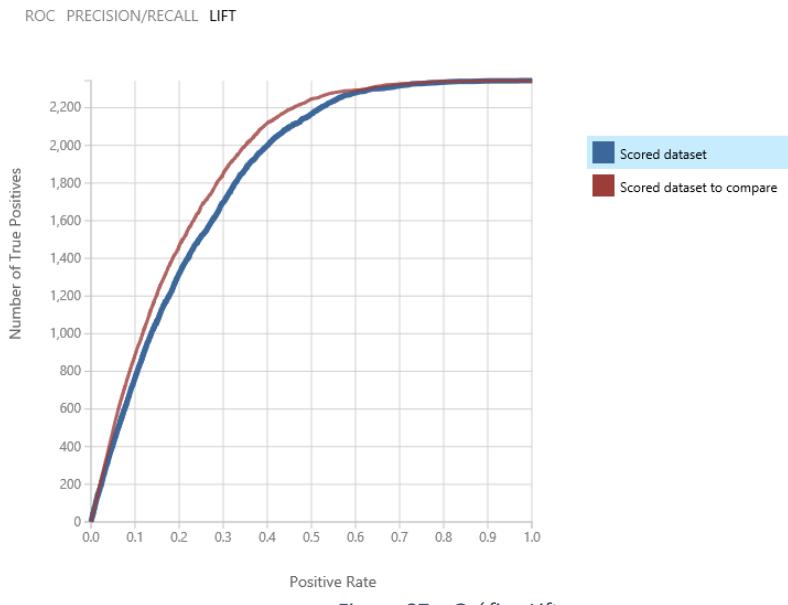


Figura 87 – Gráfico Lift

## Modelos de Regressão

Como visto em diversas partes deste livro, os problemas não se tratam apenas de classificação (ou detecção de anomalias – derivação de um problema de classificação). Outro grupo grande para validarmos os resultados são os problemas de regressão, onde o objetivo é a previsão de um valor (potencialmente infinitas saídas).

Neste caso existem as seguintes métricas que o AzureML calcula com o módulo *Evaluate Model*, conforme apresentado na Figura 88.

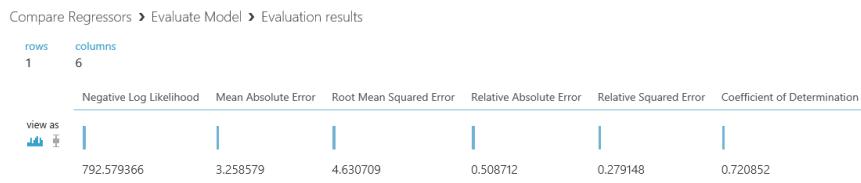


Figura 88 - Validação de modelos de regressão

- Negative Log Likelihood
- Mean Absolute Error
- Root Mean Squared Error
- Relative Absolute Error
- Relative Squared Error
- Coefficient of Determination

No cenário de aprendizagem de máquina, erro representa a diferença entre o valor previsto e o valor real. Estas medidas que envolvem “erro” no nome se diferem na escala, mas todas buscam representar o quanto distante dos valores corretos estão os valores que meu modelo forneceu.

Um exemplo simples seria o resultado calculado de 3020 quando o valor correto (conhecido antes do experimento – aprendizagem supervisionada) é 3000, neste caso o erro absoluto seria de “20”.

O valor absoluto ou o valor quadrático (*square*) desta diferença é normalmente calculado para capturar a magnitude total do erro sobre todas as instâncias (valores). Valores menores para o erro (*Mean Absolute Error*, *Root Mean Squared Error*, *Relative Absolute Error*, *Relative Squared Error*) indicam melhores com melhor acurácia na execução de previsões.

A métrica chamada coeficiente de determinação (*Coefficient of Determination*), também conhecido como R-quadrado ( $R^2$ ), é o método padrão para medir o quanto bem o modelo se adapta aos dados utilizados. É a proporção da variação explicada pelo modelo. Estes valores variam de 0 a 1, onde maiores valores são melhores. Um coeficiente de determinação igual a 1 significa o modelo perfeito. Em geral utilizamos esta métrica para comparar modelos regressivos.

## Cross Validate Model

Na seção anterior abordamos sobre validação do modelo, onde havíamos feito divisões simples dos nossos dados com o componente *Split*.

No entanto, existem outras técnicas que buscam minimizar o problema de conjuntos de treino e validação mal selecionados.

O módulo *Cross Validate Model* é a implementação de uma técnica padrão na área para verificar tanto a variabilidade quanto a confiabilidade de qualquer modelo utilizando aquele conjunto de dados. Esta é uma das principais técnicas utilizadas para validação.

O módulo recebe um modelo não treinado de classificação ou regressão e um *dataset*, conjunto de dados, como entrada. O conjunto de dados é dividido em *folds*, subconjuntos, onde para cada um é construído um modelo e também exibe um conjunto de estatísticas de acurácia.

Em outras palavras, se estamos utilizando uma divisão em 10 *folds*, teremos o modelo treinado e avaliado 10 vezes, onde para cada iteração um *fold*, subconjunto, é usado como conjunto de validação e os demais para treino.

A comparação das estatísticas de acurácia para cada *fold* permite entender como a variação dos dados afeta, e o quão suscetível é, o seu modelo.

A Figura 9 ilustra um modelo simples que utiliza a técnica de *split* para separar os conjuntos de validação e treino.

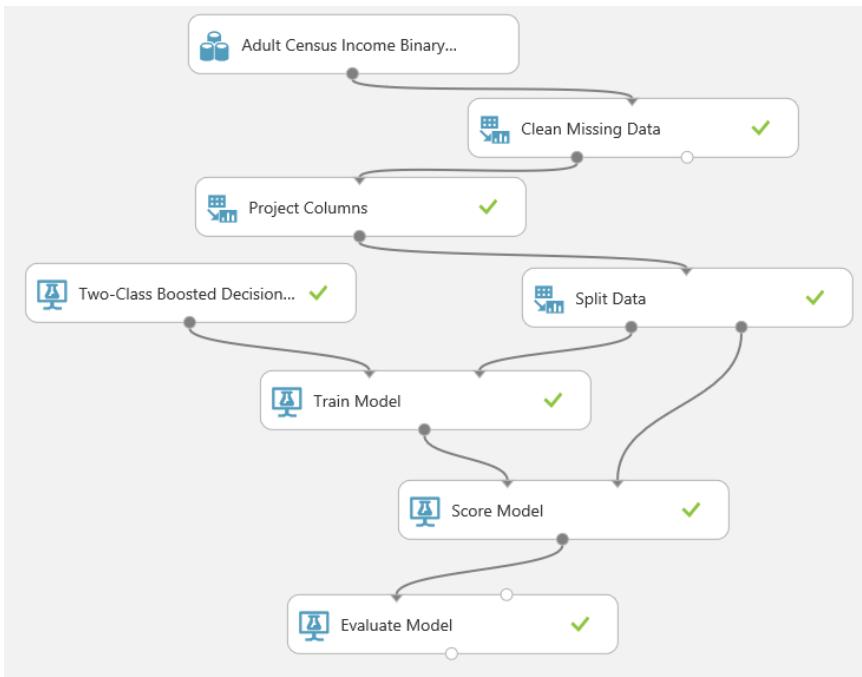


Figura 89 – Modelo utilizando o módulo Split

Os resultados do modelo ilustrado na Figura 89 não necessariamente condizem com a média de resultados que o modelo pode encontrar. Para isso lançamos mão do uso do módulo de *Cross Validate*.

Por padrão o módulo divide o seu conjunto em dez *folds* – *ten fold cross validation* – que é quantidade média de *folds* amplamente utilizada por cientistas de dados. No AzureML é possível alterar esse valor padrão através do uso do módulo *Partition and Sample*.

Os únicos parâmetros exigidos no módulo são a coluna de classe (*label*) e a semente para calcular os números aleatórios (valores iguais geram sequências de valores pseudoaleatórios iguais).

Alterando este modelo de exemplo para utilizar o módulo de validação cruzada temos o resultado mostrado na Figura 90.

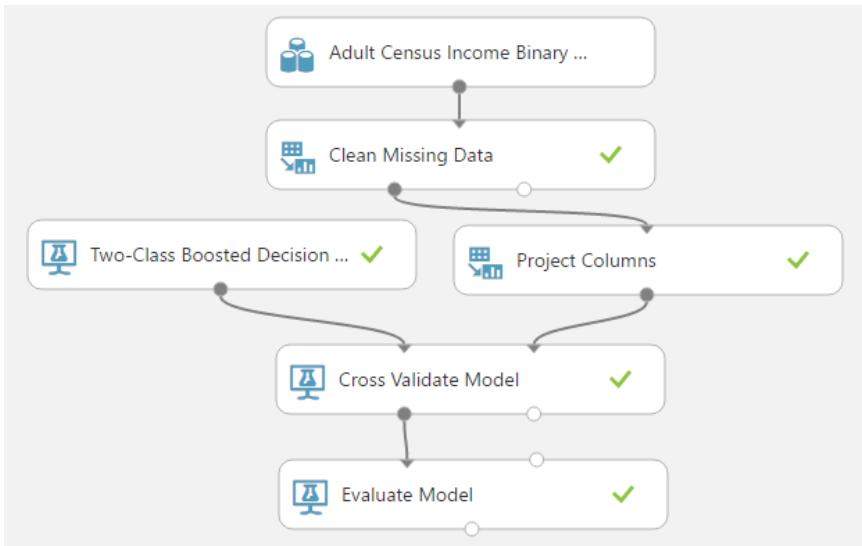


Figura 90 – Modelo utilizando o módulo *Cross Validate Model*

A Figura 81 mostra o resultado obtido ao clicar no segundo *output* do módulo, emitido por este método de validação.

Fold Number	Number of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss	Training Log Loss
0	3256	FastTree (Boosted Trees) Classification	0.869165	0.755848	0.666237	0.708219	0.92513	0.285559	47.999551
1	3256	FastTree (Boosted Trees) Classification	0.859029	0.715108	0.655673	0.684102	0.921373	0.287083	47.094931
2	3256	FastTree (Boosted Trees) Classification	0.867015	0.763348	0.662907	0.709591	0.924492	0.288713	48.154579
3	3256	FastTree (Boosted Trees) Classification	0.872543	0.753644	0.677588	0.713596	0.923332	0.284258	47.79125
4	3256	FastTree (Boosted Trees) Classification	0.869779	0.77809	0.67561	0.723238	0.930314	0.280503	50.296331
5	3256	FastTree (Boosted Trees) Classification	0.863022	0.755814	0.651629	0.699865	0.922029	0.295319	46.968224
6	3256	FastTree (Boosted Trees) Classification	0.867629	0.748851	0.646825	0.694109	0.923634	0.284989	47.409765
7	3257	FastTree (Boosted Trees) Classification	0.862757	0.752874	0.65582	0.701003	0.918376	0.299346	46.270177
8	3256	FastTree (Boosted Trees) Classification	0.858722	0.738686	0.642948	0.6875	0.918379	0.297113	46.276422
9	3256	FastTree (Boosted Trees) Classification	0.859029	0.729958	0.660305	0.693387	0.919584	0.297611	46.152145
Mean	32561	FastTree (Boosted Trees) Classification	0.864869	0.749222	0.659554	0.701461	0.922664	0.290049	47.441337
Standard Deviation	32561	FastTree (Boosted Trees) Classification	0.005028	0.017636	0.011397	0.012254	0.003616	0.006686	1.239589

Figura 91 – Saída 2 do módulo Cross Validate Model

Resultados comparativos para cada fold analisado.

Nos cenários de validação cruzada a média da acurácia (resultado da coluna *Accuracy* e linha *Mean*) e o seu desvio padrão (encontrado na coluna *Accuracy* e linha *Standard Deviation*) são em geral utilizados para explicar o modelo.

Ainda que tenha uma média de acurácia elevada, um desvio padrão elevado pode significar que o seu modelo é bastante suscetível ao conjunto de dados de entrada e deveria ser revisto.

## Módulo Partition and Sample

Como dito anteriormente, a divisão padrão do módulo *Cross Validate Model* é de dez *folds*. Para efetuarmos divisões manuais de valores diferentes, necessitamos utilizar o módulo *Partition and Sample*.

Este módulo possibilita a divisão dos dados em um número parametrizado de *folds*, além de executar a tarefa de *sampling* (amostragem). Com o *sampling* é possível selecionar apenas uma parte dos dados de entrada e ainda assim manter uma distribuição justa dos dados.

O modo de execução do módulo é definido no parâmetro *Partition or sample mode*, onde as opções são: *Sampling*, *Assign to folds*, *Pick Fold*, e *Head*.

## Sampling

Com esta opção é possível realizar dois tipos de amostragem: aleatória ou aleatória estratificada. É útil se você quer criar um conjunto de dados menor, porém representativo, para testes do seu modelo. Conjuntos de dados muito grandes levam um tempo significativo para serem treinados e validados. Uma estratégia de *sampling* pode auxiliar o cientista de dados a encontrar soluções de uma forma mais rápida.

O principal parâmetro deste modo é o *Rate of Sampling* que é a taxa de amostragem desejada. Uma taxa de 0,1 retornará 10% das linhas do *dataset*, 0,3 retornará 30% e assim por diante.

Outro parâmetro é a semente, encontrada na plataforma como *Random seed for sampling*, usada de base para o algoritmo de seleção aleatória da amostragem. É um parâmetro importante para garantir a reproduzibilidade de determinado cenário/experimento.

Por fim, temos *Stratified split for sampling*, cuja seleção indica se desejamos que os resultados sejam estratificados, baseados em uma coluna selecionada, ou não.

Imagine que o desejado é uma amostragem para um problema de classificação, teríamos problemas caso algumas classes possuam muito mais exemplos que outras. Neste caso uma amostragem estratificada baseada na coluna de classe irá garantir que para cada classe seja adicionado ao conjunto final na proporção definida pelo *Rate of Sampling*.

Em outras palavras, uma proporção de 0,1 com a coluna de classe selecionada irá criar um conjunto final contendo 10% das linhas de cada classe selecionadas aleatoriamente, baseado na semente, dentro das mesmas.

## Assign to Folds

Este modo serve para dividir o *dataset* em um dado número de *folds*.

Para o cenário de divisão em *folds* similar ao encontrado no módulo *Cross Validate Model*, é importante se atentar ao parâmetro *Use replacement in the partitioning*. Este parâmetro, se verdadeiro, permite que uma determinada linha que já tenha sido usada em um determinado *fold*, possa ser novamente utilizada em um novo. Para garantir a validação cruzada é importante que um mesmo exemplo não apareça ao mesmo tempo no conjunto de validação e treino.

O segundo parâmetro importante neste modo é o método de particionamento, encontrado como *specify the partitioner method*, que determina como os *folds* serão montados.

*Partition evenly*: Teremos um número igual de valores/linhas em cada *fold/partição*.

*Partition with customized proportions*: Pode especificar uma lista, separada por vírgula, com as proporções para cada *fold*.

Por exemplo, a geração de 4 subconjuntos, onde a distribuições dos registros seja 40% para o primeiro, 20% para o segundo, 30% para o terceiro e 10% para o quarto subconjunto. Então devemos especificar: **0.4, 0.2, 0.3, 0.1**.

## Pick Fold

Com este modo podemos especificar como retorno um dos *folds* criados previamente no modo *Assign to Folds*.

Temos dois parâmetros neste cenário: *Specify which fold to be sampled from* e *Pick complement of the selected fold*.

*Specify which fold to be sampled from:* Neste parâmetro podemos especificar o índice do *fold* que desejamos retornar. Este índice se inicia em 1 (no nosso exemplo com quatro *folds*, teríamos a possibilidade de fornecer os valores de 1 a 4).

No caso da seleção do valor de um *fold* inexistente será emitido um erro<sup>15</sup>.

*Pick complement of the selected fold:* Caso marcado, especifica que o desejado como retorno é o complemento do *fold* definido no parâmetro anterior.

No nosso exemplo anterior (40%, 20%, 30%, 10%), ao especificar o **valor 2 para specify which fold to be sampled from** e marcar a caixa de seleção *pick complement of the selected fold* teremos como resultado 80% dos dados, sendo equivalente aos registros contidos nos subconjuntos 1, 3 e 4.

## Head

Este modo retorna os N primeiros registros do *dataset*. Caso o número especificado seja maior do que o tamanho do dataset, serão retornados todos os registros do dataset.

A Figura 92 ilustra o uso deste módulo para *sampling*, *assign to fold* e *pick fold*.

---

*15 Error 0018: Dataset contains invalid data*

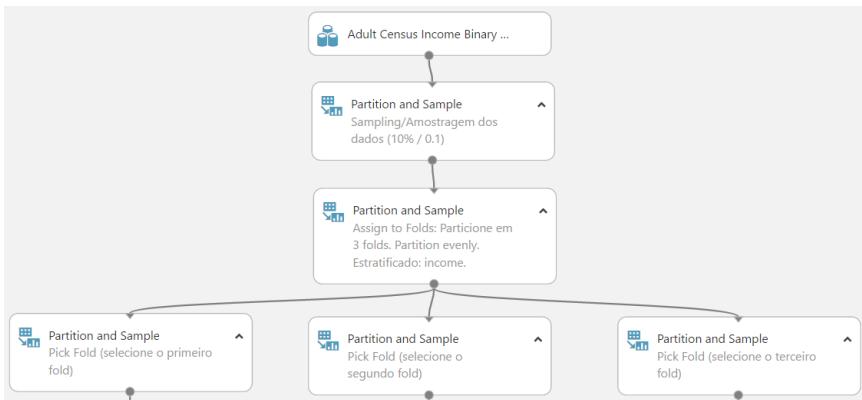


Figura 92 – Exemplo da utilização de módulo Partition and Sample

Como vimos ao longo do capítulo, existem diversas métricas e formas diferentes de validar o seu modelo. Esta escolha é influenciada pelo formato do problema (classificação, regressão, etc.), pelo seu volume (técnicas de amostragem), e também pela necessidade da validação de quão suscetível o modelo é quando temos a variação dos dados de entrada.

Não há fórmula mágica de decisão e cada situação pode exigir investir mais tempo em uma métrica ou outra, em uma técnica ou outra. Cabe ao cientista de dados fazer experimentação e usar a sua experiência para determinar o melhor caminho (ou ao menos os caminhos possíveis) para os cenários estudados ou avaliados.

## 6 – Expondo Modelos Através de Web Services

### Introdução e publicação dos web services do Azure Machine Learning

A criação de modelos desenvolvidos no AzureML é totalmente realizada na web. Desta forma o seu consumo também é baseado na web, através de *web services*.

Um *web service* é uma forma de integração entre aplicações diferentes, bem como a sua comunicação, baseado em chamadas *http*, sigla de *Hypertext Transfer Protocol*. Isso significa que o acesso será possível por qualquer aplicação codificada em uma linguagem capaz de entendê-los. Todas as principais linguagens modernas de programação suportam chamadas à *web services*.

Para adicionar um *web service* a um modelo já criado é necessário a configuração dos módulos de *Input* e *Output* de *Web Service*, e após isso é que devemos seguir para a sua publicação.

A adição dos módulos pode ser feita diretamente através dos módulos da interface da toolbox, como apresentado na Figura 93, ou através do botão *Setup Web Service*, como feito na Figura 94. O botão de setup adiciona o *Input* e *Output* no início e final do experimento, mas você pode ajustar conforme a sua necessidade, observe os comentários referentes à Figura 96.

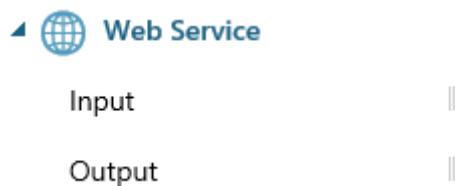


Figura 93 – Módulos relativos ao webservice



Figura 94 – Set Up Web Service

Observe na Figura 95 um exemplo de experimento que possui estes módulos configurados.

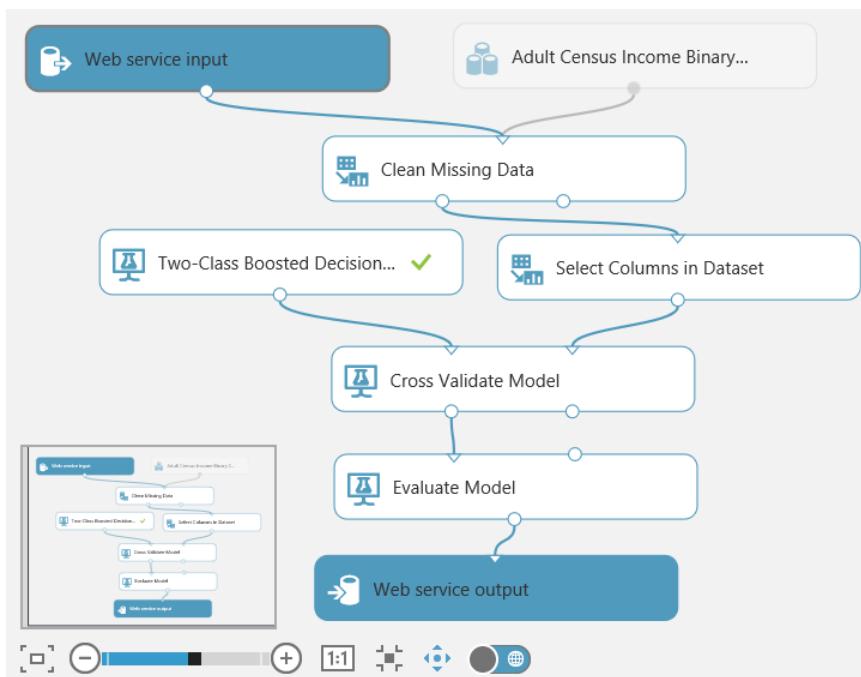


Figura 95 – Exemplo de experimento com os webservices após clicar no botão Set Up Web Service.

É importante sabermos que a posição onde um módulo destes se encontra influencia os parâmetros que serão necessários durante o seu consumo. Isto vale tanto nos valores que serão configurados como entrada, bem como a saída/resultado. Os elementos necessários e os elementos de saída respeitam as mesmas características do local do fluxo ao qual estão conectados.

Observe na Figura 95 que o módulo *Web service input* está conectado no mesmo local que o *Dataset* chamado *Adult Census Income Binary...*, isto fará com que quem consuma este *web service* precise fornecer dados com a mesma estrutura que o conjunto de dados possui. De maneira similar, a saída deste *web service* será a saída do módulo *Evaluate Model*.

Durante a adição dos referidos módulos é possível observar que é adicionado na interface gráfica um botão para escolher o modo de execução do experimento. Isto é útil para a sua validação, pois alterna entre o modo utilizando os módulos de *web services*, através do *input* e *output* ou o modo utilizando o conjunto de dados presente no experimento.

De forma intencional foram adicionados os módulos de *web services* ao experimento da Figura 95. Avalie que neste experimento é onde criamos, também usamos o termo treinamos, o modelo e não é a maneira correta para seu consumo, e sim para sua avaliação (experimentação incremental).

Para o consumo devemos ter um experimento apartado, conhecido em geral como experimento preditivo, também pode ser chamado de experimento de *scoring*. E este é quem deve conter os módulos que apresentamos. Perceba na Figura 96 que não há treino neste experimento, utilizamos apenas o modelo já treinado. O ideal é que este experimento preditivo execute rapidamente, justamente por não conter treino, apenas a classificação direta, conhecida como *scoring*.

Observe na Figura 96 um exemplo correto da utilização dos mesmos. Perceba a utilização do módulo *Select Columns in Dataset*, que anteriormente era conhecido como *Project Columns*, entre o *Score Model* e o *Web Service output*. Isto serve como técnica para reduzir o número de colunas retornadas pelo *web service*.

Você deve estar se perguntando o porquê de um experimento de predição precisar ter o conjunto de dados original anexado a ele. É verdade que este experimento não precisará dos dados, contudo precisará dos metadados, informação sobre as colunas dos dados. É por isso que necessitamos tê-lo junto ao fluxo.

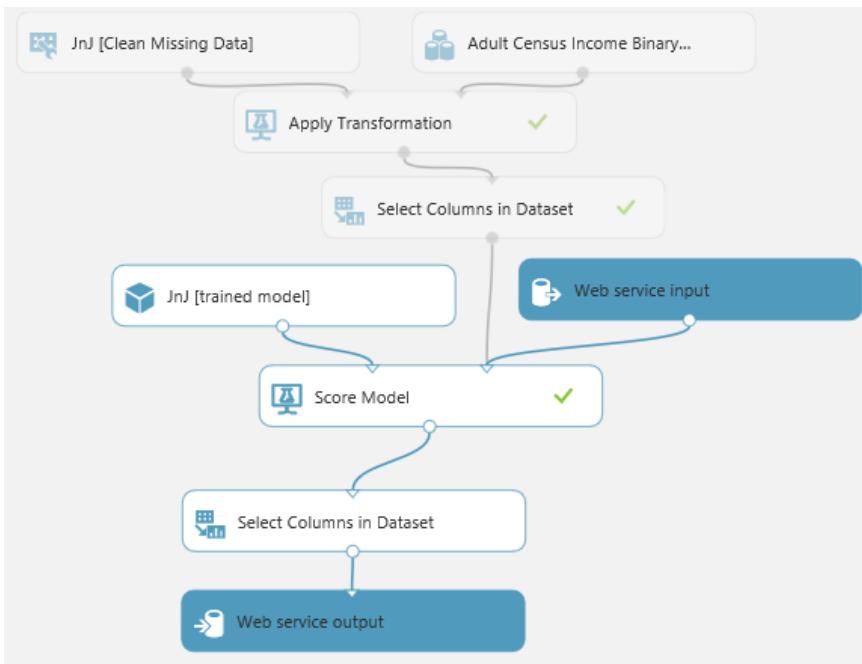


Figura 96 – Experimento preditivo utilizando web services

Conforme dito anteriormente, após a adição destes módulos é necessária a publicação do *web service*, para isso basta clicar no botão *Deploy Web Service* localizado na barra inferior, conforme mostra a Figura 97. Será necessário executar o experimento através do click no botão *Run* antes desta ação.



Figura 97 – Botão Deploy Web Service

Após a implantação o *web service* aparecerá listado no grupo de *web services* publicados no seu *Azure Machine Learning*, conforme mostra a Figura 98.

A screenshot of the 'published web services' list in the Azure Machine Learning studio. On the left, there is a sidebar with icons for 'PROJECTS', 'EXPERIMENTS', and 'WEB SERVICES'. The 'WEB SERVICES' section is selected. To the right, the 'published web services' list is shown. It has a header 'NAME' and two entries: 'Sample [Predictive Exp.]' and 'Experiment created on 11/13/2015 [Predictive Exp.]'. The first entry is highlighted with a blue background.

Figura 98 – Web Services publicados

## Configuração dos web services publicados

Após a publicação, também conhecido como *deploy*, do *web service* é necessária a sua configuração através do painel de *web services* publicados.

As informações ficam divididas em duas abas: *dashboard* e *configuration*.

### Aba Dashboard

A aba de *dashboard* fornece informações sobre o respectivo *web service*, bem como a chave de segurança, encontrada com o nome de *API Key*, utilizada na chamada dos mesmos. Também apresenta algumas formas para validar o modelo e informações para adicionar *endpoints* adicionais. A adição de novos *endpoints* é realizada através do portal de gerenciamento do Azure,

e a sua utilização é recomendada para ambientes com grande utilização. A Figura 99 ilustra a aba *dashboard*.

sample [predictive exp.]

The screenshot shows the 'DASHBOARD' tab selected in the top navigation bar. Below it, there's a 'CONFIGURATION' section with tabs for 'General', 'Published experiment', 'View snapshot', and 'View latest'. Under 'General', there's a 'Description' field containing the placeholder 'No description provided for this web service.' and an 'API key' input field with a value starting with 'qXEGgpCRLt7CZqc2XpYb7aCQbCvQiNH+bKLGnoh+s6KWDP+B1Ri4'. Below this is a 'Default Endpoint' section with tabs for 'API HELP PAGE', 'TEST' (which is highlighted in blue), and 'APPS'. The 'TEST' tab shows two entries: 'REQUEST/RESPONSE' and 'BATCH EXECUTION'. The 'REQUEST/RESPONSE' entry has a status of 'Excel 2013 or later | Excel 2' and was last updated on '5/16/2016 2:00:17 PM'. The 'BATCH EXECUTION' entry has a status of 'Excel 2013 or later workbook' and was last updated on '5/16/2016 2:00:17 PM'. At the bottom, there's a section for 'Additional endpoints' with a note that 'Number of additional endpoints created for this web service: 0' and a link to 'Manage endpoints in Azure management portal'.

Figura 99 – Aba Dashboard (Web Service)

O teste do modelo pode ocorrer ao clicar no botão *Test*, onde será aberto uma janela com os dados necessários para executar o experimento, conforme apresentado na Figura 100. Este formato permite a entrada de apenas um elemento para teste por vez. O Excel também pode ser utilizado para validação.

Ao clicar nos botões referentes ao Excel, uma planilha com o *add-in* do *Azure Machine Learning* será aberta, acompanhe esta ação na Figura 101, já contendo todos os dados necessários para a conexão com este *web service*. Nas versões anteriores era utilizada uma planilha com macro, formato .xlsm, porém a mesma foi substituída para este formato apresentado.

Test Sample [Predictive Exp.] Service

## Enter data to predict

AGE

EDUCATION

EDUCATION-NUM

MARITAL-STATUS

RELATIONSHIP

Figura 100 – Teste do modelo através da janela de teste

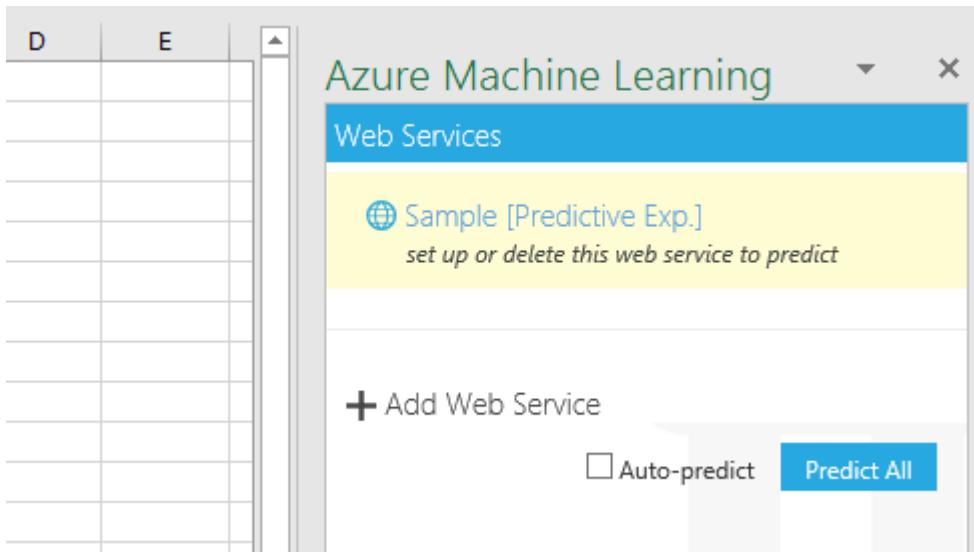


Figura 101 – Planilha para teste do web service (add-in do Azure Machine Learning)

## Aba Configuration

Na aba de configuração é possível configurar o nome apresentado para este *web service*, bem como sua descrição.

Também estão disponíveis as configurações sobre log dos acessos ao *web service*, encontrado como *Enable Logging*, que vem desativado por padrão, e a possibilidade de uso de dados de exemplo, encontrado como *Enable Sample Data?*, que é ativo por padrão.

É possível também a configuração de rótulos, encontrados como *labels*, amigáveis para os parâmetros que o *web service* recebe e também para os parâmetros de saída.

A Figura 102 retrata a aba de configuração, encontrada no portal como *Configuration*.

## sample [predictive exp.]

DASHBOARD    CONFIGURATION

### settings

#### GENERAL

Display Name	Sample [Predictive Exp.]
Description	No description provided for this web service.

#### ENABLE LOGGING

YES    **NO**    [Learn more](#)

#### ENABLE SAMPLE DATA?

**YES**    NO

#### INPUT SCHEMA

age (Numeric)	idade
---------------	-------

**N**      **SAVE**      **DISCARD**

Figura 102 – Aba de configurações (Web Service)

Conforme vimos, o *AzureML Studio* fornece a possibilidade de testarmos nossos modelos através de si próprio ou através do Microsoft Excel. Contudo, no mundo real precisamos integrar a comunicação com o *AzureML* em aplicações próprias.

O mecanismo para comunicação é baseado em chamadas *HTTP*, protocolo *SOAP* sobre *HTTP*, ou seja, quaisquer linguagens de programação que suportem essa integração podem ser utilizadas para tal. Podemos citar R, C# (.NET), Java, Python, etc. Maiores detalhes e exemplos de código para

consumo através de aplicações podem ser encontradas na documentação oficial do AzureML online<sup>16</sup>.

## Alterando os Parâmetros do Web Service

Em alguns casos existe a necessidade de alterar o comportamento de um módulo durante a sua execução. Para os cenários que isso é necessário existe um recurso chamado *Web Services Parameters*.

Alguns cenários comuns que podem necessitar de alteração durante a execução é o módulo *Import Data*, anteriormente conhecido como *Reader*, para ler de uma fonte diferente, ou do módulo *Export Data*, anteriormente conhecido como *Writer*, para escrever em uma fonte diferente, entre outros.

Imagine que desejamos incrementar nosso modelo apresentado na Figura 96. Para efeitos de demonstração do recurso, vamos salvar o output em um *Azure Blob Storage*<sup>17</sup>. Adicionamos um módulo de *Export Data* na saída do módulo *Select Columns in Dataset*.

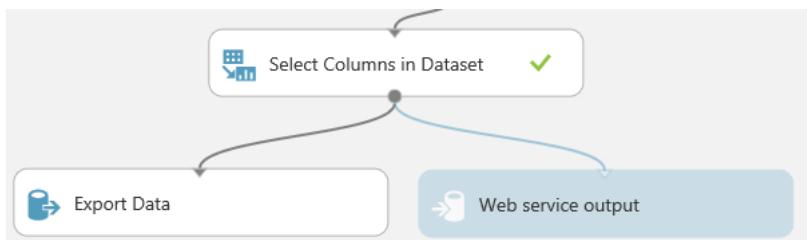


Figura 103 – Experimento preditivo com o módulo Export Data

Após isso é necessário configurar o mesmo. Neste nosso exemplo vou utilizar um *blob* no *Microsoft Azure*, veja mais sobre como trabalhar com dados

16 <https://azure.microsoft.com/en-gb/documentation/articles/machine-learning-connect-to-azure-machine-learning-web-service/>.

17 <https://azure.microsoft.com/en-us/services/storage/blobs/>

externos no Capítulo 3 – Trabalhando com Dados Externos. Para reproduzir este exemplo você necessitará de uma *storage account* e um *container* criados no *Microsoft Azure*. Maiores informações de como fazer isso podem ser encontradas na documentação online<sup>18</sup>.

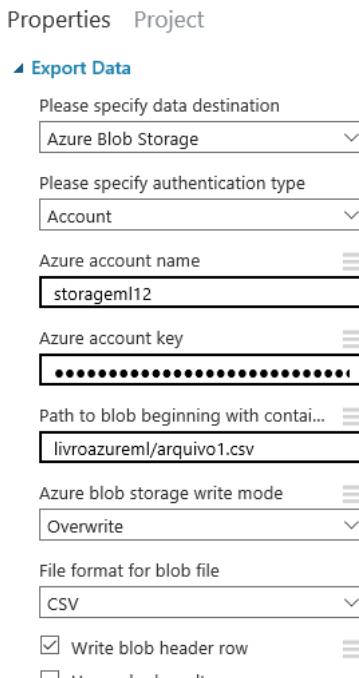


Figura 104 – Propriedades do módulo Export Data

Após a execução um arquivo chamado arquivo1.csv é gerado no *container* de nome livroazureml.

Imaginem que o desejado é ter integração do modelo desenvolvido em uma aplicação através de uma comunicação via web services, mas que seja

---

<sup>18</sup> <https://azure.microsoft.com/en-us/documentation/articles/storage-create-storage-account/>

necessário gerar um arquivo diferente para cada execução para propósitos de log. Um exemplo para esta necessidade é gerar logs e analisar em batch no futuro. Para isso podemos configurar o parâmetro Path to blob beginning with container como um parâmetro do web service, conforme ilustrado na Figura 105, e com isso ter a capacidade de sobrescrevê-lo (o valor do parâmetro) em tempo de execução. Para cada nova execução podemos fornecer um valor diferente para este parâmetro.

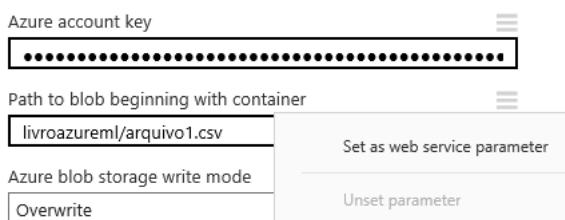


Figura 105 – Configuração de propriedade como um parâmetro do web service

Depois desta configuração, precisamos salvar e executar o experimento e depois fazer o deploy deste *web service*. Um ícone de globo aparecerá ao lado do nome do parâmetro para indicar que é uma propriedade associada a um parâmetro.

Na aba de *web services* publicados, podemos testá-lo passando um valor para o parâmetro diferente do originalmente utilizado. Repare que agora há uma opção para informar o valor da propriedade de destino do arquivo. No exemplo, mostrado na Figura 106, estamos utilizando como destino o arquivo *arquivo\_dinamico.csv* e o colocaremos no mesmo container do arquivo originalmente configurado.

Test Sample [Predictive Exp.] Service

HOURS-PER-WEEK

NATIVE-COUNTRY

INCOME

Enter Web Service Parameters

PATH TO BLOB BEGINNING WITH CONTAINER

livroazureml/arquivo\_dinamico.csv

✓

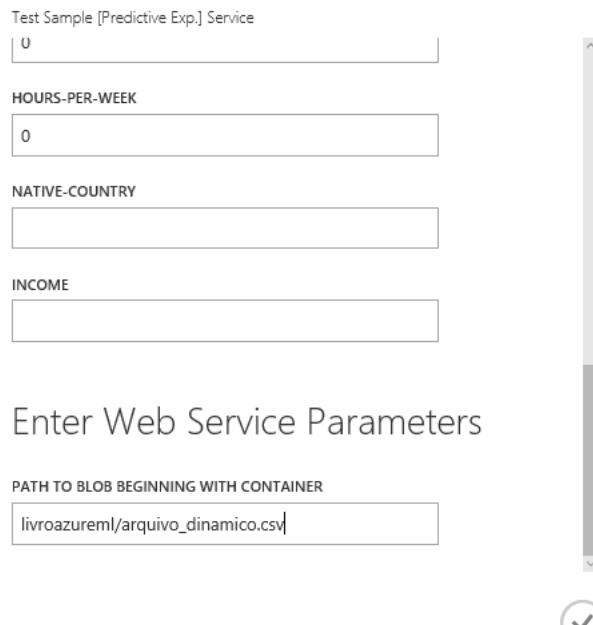


Figura 106 – Alterando o parâmetro path to blob beggining with container dinamicamente

Como resultado temos a geração do arquivo desejado, conforme pode ser visto na Figura 107 que retrata os objetos presentes no *container* especificado.

The screenshot shows the Azure Storage Explorer interface. At the top, there's a dark blue header bar with the text "livroazureml" and "Container". Below the header are four icons: Refresh, Delete, Properties, and Access policy. A search bar at the top says "Search blobs by prefix (case-sensitive)". The main area displays a table of blobs:

NAME	MODIFIED	BLOB TYPE	SIZE
arquivo_dinamico.csv	5/28/2016 6:04:30 PM	Block blob	41 B
arquivo1.csv	5/28/2016 5:56:36 PM	Block blob	814.97 KB

*Figura 107 – Arquivos gerados no container livroazureml*

## APIs de Retreino – Retraining APIs

Existem cenários cujas fontes dos modelos mudam rapidamente e/ou que é necessária o seu retreino de forma automatizada. Em outras palavras, reprocessamento do modelo com novos dados e sem, ou com baixa, intervenção humana. Nem sempre isto é uma necessidade, mas vejamos o exemplo a seguir.

Imagine que o seu modelo faça a sugestão de produtos para visitantes em seu e-commerce baseado no histórico de navegação e compra de produtos. Também suponha que no ramo do seu e-commerce a sua base de produtos ofertados cresça regularmente. Neste caso uma das necessidades é incluir informações sobre os novos produtos dentro do seu modelo, bem como pode ser necessário deixar de considerar produtos obsoletos ou que não sejam mais fabricados. Por exemplo, fazer retreino semanal, diário ou com alguma periodicidade própria.

Para isto existe a API de retreino do modelo, também baseada em web services.

É importante mencionar que para funcionar como o esperado devemos supor que atendemos às seguintes observações:

1. A fonte de dados é alguma fonte capaz de perceber as mudanças realizadas nos dados. Para isso pode ser uma database propriamente dita (*Azure SQL Database*), uma outra fonte que passe por atualização, como por exemplo um novo arquivo em um *blob* no *Microsoft Azure*. O uso de parâmetros alterados em tempo de execução, como vimos na seção anterior, pode ajudar a apontar seu experimento para outras fontes.
2. Existe algum mecanismo de agendamento externo que executa a chamada ao web service na periodicidade desejada como o *task scheduler* do *Windows*, *Azure Data Factory*, etc. O Azure Machine Learning não contém mecanismo de agendamento automatizado.

Para retreinar o modelo, você precisa fazer o *deploy* do experimento de treino que você criou como um web service do tipo retreino. Este web service necessita de um módulo de *Web Service Output* conectado na saída do módulo *Train Model* para produzir novos modelos treinados. Também é possível utilizar um módulo de saída de web service para o módulo *Evaluate Model*, e assim ter as informações sobre desempenho do modelo retreinado.

Observe na Figura 108 um exemplo treinado. Perceba os módulos de *web services* adicionados e também que já executamos o experimento, esta execução é necessária para habilitar a publicação.

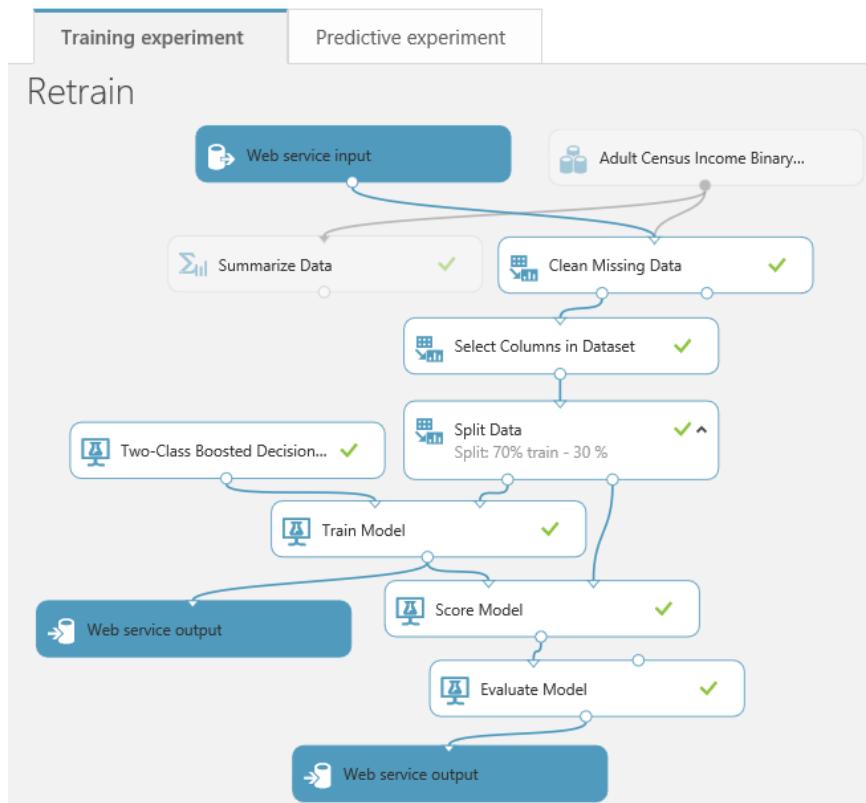


Figura 108 – Exemplo de experimento com módulos para retreino

Após adicionar os módulos e executar o seu experimento, faça o deploy utilizando o botão *SET UP WEB SERVICE* e então *Deploy Web Service [Classic]*, com isso teremos exposto o modelo para retreino. Na versão do momento da publicação deste livro, o único método suportado para o retreino é através da API de *BATCH*.

Um exemplo completo utilizando C# para retreinar o modelo pode ser encontrado na documentação oficial<sup>19</sup>.

---

<sup>19</sup> <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-retrain-models-programmatically/#retrain-the-model-with-new-data-using-bes>.

## 7 – Introdução à Algoritmos de Machine Learning

Utilizar o algoritmo que melhor responde ao problema é uma tarefa primordial dentro do processo de Aprendizagem de Máquina. Não existe o modelo certo ou errado, existe o modelo que apresenta o melhor resultado para o problema estudado. Para isso, é importante conhecer os algoritmos e seus respectivos objetivos, e é disso que trata este capítulo. Aqui serão abordados os principais algoritmos existentes no *Azure Machine Learning* e como eles funcionam internamente. Ao conhecer o grupo ao qual o algoritmo faz parte, fica mais fácil saber quais são os processos existentes para aquela solução.

### Introdução aos grupos de algoritmos

Conhecer os grupos dos quais os algoritmos fazem parte, facilita a busca do algoritmo mais apropriado a responder de forma eficiente ao problema. Abaixo estão as categorias de algoritmos existentes no *Azure Machine Learning* no momento que escrevemos este livro.

**Classificação:** Algoritmos de Classificação permitem que análises sejam feitas nas bases de treino e teste, e seu valor preditivo, aquele que estamos buscando, possa ser encontrado de forma discreta. Um resultado discreto significa apresentar um item baseado em um subconjunto conhecido de valores possíveis. Normalmente são poucos valores. Os mais comuns de se encontrar no mercado são classificadores binários, que fazem a predição da resposta ser SIM ou NÃO, verdadeiro ou falso – zero ou um – ou qualquer outra diferenciação de apenas duas possibilidades (classes), e apresentam o grau do índice de acurácia, também chamado de confiança em algumas literaturas, para aquela resposta que foi apresentada.

Em diversos cenários uma classificação binária pode não resolver o problema, por existirem mais do que duas possíveis respostas para aquele problema, neste caso, a classificação é conhecida como Multi-Classes.

Um exemplo que utiliza este processo pode ser uma análise que identifica indivíduos em grupos de cargos dentro da empresa, onde as variáveis são analisadas e um padrão de respostas é encontrado em cada grupo já conhecido. Isso faz com que aquele conjunto discreto de possibilidades seja buscado automaticamente e encontrado sempre que um funcionário se candidatar a uma vaga. Se o candidato se candidata para uma posição de gestão, mas a classificação o mantém em um resultado técnico, pode ser que não seja uma boa opção fazer a contratação ou promoção daquele indivíduo porque suas habilidades são mais direcionadas para a área técnica do que de gestão.

Uma subcategoria dos problemas de classificação seria a de detecção de anomalias. Nestes casos o problema é de classificação, onde o objetivo é a detecção de valores anômalos (*outliers*). Nestes casos é comum termos poucos valores sobre uma das classes desejadas.

Imagine um cenário de detecção de fraudes bancárias. As classes possíveis para uma transação financeira seriam: fraude ou não fraude. Exemplos da classe de fraude são escassos comparados aos exemplos de não fraude, e isso pode causar resultados pobres se utilizarmos as técnicas de classificação tradicionais de duas classes. Para estes casos existem algoritmos que funcionam de forma mais adequada, conhecidos como classificadores de uma classe e que abordaremos mais adiante neste mesmo capítulo.

**Régressão:** Algoritmos de Regressão permitem que as análises na base de treino e testes retornem valores contínuos para aquele resultado preditivo. Por valores contínuos entendemos que são valores quase que “infinitos”, podendo ser qualquer valor possível dentro de um intervalo. Uma aplicação que utiliza este grupo de algoritmo geralmente utiliza variáveis dependentes e independentes.

Um exemplo que utiliza este grupo de algoritmo pode ser a especificação de um imóvel. Uma resposta deste processo pode variar em intervalos com valores entre 1 dólar como as casas em Detroit depois do

problema da bolha imobiliária<sup>20</sup> e 380.000.000 dólares como uma cobertura no Principado de Mônaco<sup>21</sup>. Este intervalo de possibilidades é enorme, e uma aplicação que faça este tipo de análise pode ter uma carga de processamento e consumo de memória grandes. O uso de algoritmos de regressão para estas aplicações facilita muito o nosso trabalho! Observando os dados existentes nestas análises, o valor do imóvel é considerado a variável dependente e todos os outros que são utilizados para definir o imóvel como metragem do terreno, área construída, quantidade de cômodos, localização, e algumas outras pode ser entendido como variáveis independentes (*features / características*).

**Clusterização:** Algoritmos de *Clustering*, ou Segmentação, fazem o processo de particionar os dados da amostra em vários subconjuntos, deixando as ocorrências mais semelhantes dentro de um mesmo grupo. Diferente dos algoritmos de classificação na qual as classificações geralmente são conhecidas, em algoritmos de Clusters as observações, encontrados também como linhas ou registros, são agrupados pelas suas semelhanças.

Imaginando a aplicação deste algoritmo em um universo de pessoas, é possível segmentar estes indivíduos por região que moram, região que trabalham, sexo, faixa etária, faixa de renda, etc. Qualquer dado conhecido pode ser usado para fazer a segmentação desta amostra e a partir desta segmentação é possível criar várias outras análises.

---

20 Preço de uma casa em Detroit: <http://www.nydailynews.com/lifestyle/real-estate/1-buy-house-detroit-article-1.1415014>

21 Preço de uma casa em Mônaco: <http://gallivantguide.com/monaco-penthouse-387m-worlds-most-expensive/2098/>

## Modelos de Regressão

Completando a explicação do início do capítulo, modelos de regressão são artefatos estatísticos que resolvem modelos de previsão baseados em variáveis dependentes e independentes.

Seguindo o exemplo de precificação do imóvel, podemos definir o que são variáveis dependentes e independentes da seguinte forma:

**Dependente:** A sugestão do nome é perfeita. Esta variável depende de uma ou mais variáveis existentes no modelo. O resultado desta variável é a predição que se busca ao usar esta técnica. No exemplo da precificação de imóveis, a variável dependente é o preço, que terá seu resultado encontrado a partir das variáveis independentes, explicadas abaixo;

**Independente:** Ao contrário da variável dependente, este grupo de variáveis não dependem de outras para ter seu valor definido. Outro ponto de diferença é onde se usa esta variável, ela não é a saída da predição e sim, a entrada. É com base nestas variáveis que a regressão irá predizer o valor da variável dependente. Voltando ao nosso exemplo, a quantidade de cômodos, área construída, CEP, ou localização, área do terreno, etc. Estes elementos são invariáveis na nossa predição, porém eles influenciam diretamente o preço do imóvel que estamos buscando. As variáveis de entrada também são conhecidas como características ou no inglês, *features*. Você vai encontrar estes nomes na literatura acadêmica.

Na sequência, faremos a explicação de tipos de regressão que são comuns aos modelos preditivos.

## Regressão linear

Talvez o modelo de regressão mais comum de se encontrar no mercado de trabalho e nas pesquisas acadêmicas. A regressão linear tem por objetivo encontrar a correlação entre ao menos duas variáveis, sendo uma dependente e outra independente. Em cenários mais simples, é possível

utilizar a equação genérica abaixo para criar uma regressão que atenda à demanda de duas variáveis.

$$h\theta(x) = \theta_0 + \theta_1 x$$

Onde  $h\theta(x)$  é a hipótese (que será o valor de Y, a variável dependente) em um determinado x.

$\theta_0$  é o valor da primeira variável de entrada do modelo, e  $\theta_1 x$  é o valor da segunda variável de entrada do modelo multiplicado por x.

Esta equação apresenta uma hipótese. Veja nos próximos três exemplos, esta equação com dados e como fica o resultado. Aplicando um universo onde o X (variável independente) possui os valores 0, 1, 2 e 3.

O processo implica na execução da equação cada um dos valores da variável de entrada, variável independente, para criar a regressão linear que será utilizada para encontrar o valor de Y, variável dependente.

Exemplo 1, onde  $\theta_0 = 1,5$  e  $\theta_1 = 0$

Aplicando a equação genérica com os dados do exemplo 1, pode-se seguir como estes passos apresentados:

$$X = 0$$

$$h\theta(0) = \theta_0 + \theta_1 0$$

$$h\theta(0) = 1,5 + 0 * 0$$

$$h\theta(0) = 1,5$$

$$X = 1$$

$$h\theta(1) = \theta_0 + \theta_1 1$$

$$h\theta(1) = 1,5 + 0 * 1$$

$$h\theta(1) = 1,5$$

$$X = 2$$

$$X = 3$$

$$h\theta(2) = \theta_0 + \theta_1 \cdot 2$$

$$h\theta(2) = 1,5 + 0 \cdot 2$$

$$h\theta(2) = 1,5$$

$$h\theta(3) = \theta_0 + \theta_1 \cdot 3$$

$$h\theta(3) = 1,5 + 0 \cdot 3$$

$$h\theta(3) = 1,5$$

Plotando os gráficos em um plano cartesiano bidimensional é possível encontrar uma regressão linear simples como apresentado abaixo na Figura 109.

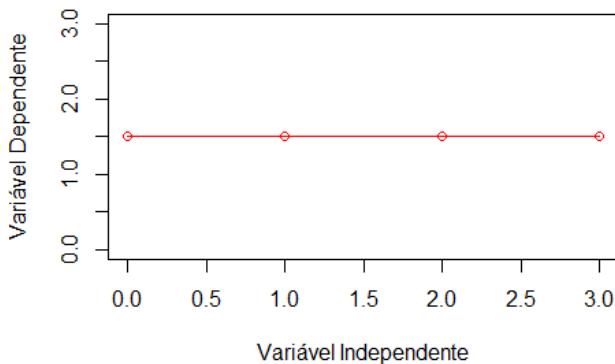


Figura 109 - Hipótese com 1,5 e 0

Exemplo 2, onde  $\theta_0 = 0$  e  $\theta_1 = 0,5$

$$X = 0$$

$$h\theta(0) = \theta_0 + \theta_1 \cdot 0$$

$$h\theta(0) = 0 + 0,5 \cdot 0$$

$$X = 1$$

$$h\theta(1) = \theta_0 + \theta_1 \cdot 1$$

$$h\theta(1) = 0 + 0,5 \cdot 1$$

$$h\theta(0) = 0$$

$$h\theta(1) = 0,5$$

$$X = 2$$

$$h\theta(2) = \theta_0 + \theta_1 \cdot 2$$

$$h\theta(2) = 0 + 0,5 \cdot 2$$

$$h\theta(2) = 1$$

$$X = 3$$

$$h\theta(3) = \theta_0 + \theta_1 \cdot 3$$

$$h\theta(3) = 0 + 0,5 \cdot 3$$

$$h\theta(3) = 1,5$$

O resultado deste cálculo nos apresenta uma reta pouco inclinada como apresentada na Figura 110.

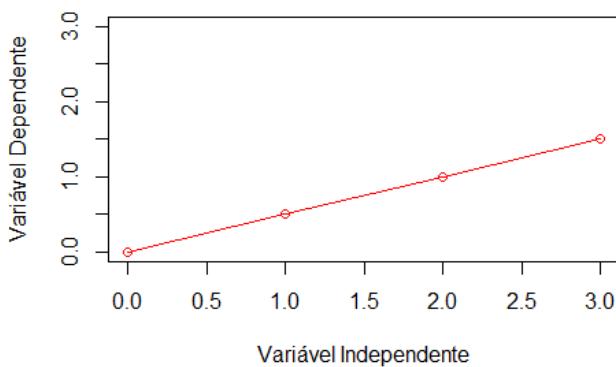


Figura 110 - Hipótese com 0 e 0,5

Exemplo 3, onde  $\theta_0 = 1$  e  $\theta_1 = 0,5$

$X=0$	$X=1$
$h\theta(0) = \theta_0 + \theta_1 0$	$h\theta(1) = \theta_0 + \theta_1 1$
$h\theta(0) = 1 + 0,5 * 0$	$h\theta(1) = 1 + 0,5 * 1$
$h\theta(0) = 1$	$h\theta(1) = 1,5$
$X=2$	$X=3$
$h\theta(2) = \theta_0 + \theta_1 2$	$h\theta(3) = \theta_0 + \theta_1 3$
$h\theta(2) = 1 + 0,5 * 2$	$h\theta(3) = 1 + 0,5 * 3$
$h\theta(2) = 2$	$h\theta(3) = 2,5$

O terceiro exemplo gera uma outra inclinação e valores dos eixos representados na Figura 111.

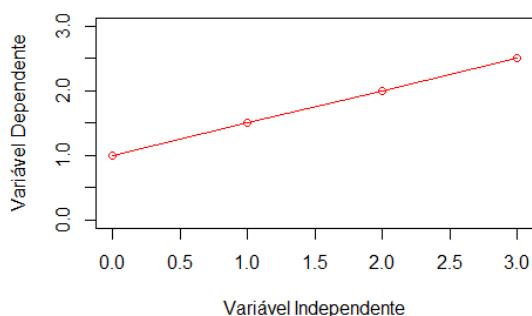


Figura 111 - Hipótese com 1 e 1,5

Entender o processo simples é necessário para aplicar este modelo de regressão com cenários complexos contendo muito mais valores. Ao invés de ter apenas quatro valores de uma única dimensão para colocar no eixo X do plano cartesiano bidimensional, imagine agora um hiperplano com muitas dimensões. Sim, é bem difícil de imaginar este hiperplano contendo diversas variáveis independentes. Para facilitar, imagine cada uma destas dimensões como sendo uma daquelas variáveis que será usada na especificação do imóvel. Uma dimensão pode ser a quantidade de cômodos, outra a metragem, outra a localização, e assim por diante. Isso é conhecido como Regressão Linear Multivariável. A complexidade matemática começa a assustar quem não é familiarizado com esta ciência, e também não é o objetivo do livro entrar na parte teórica, matemática e estatística do algoritmo. Este capítulo tem o foco em apenas mostrar como o algoritmo funciona para ajudar na escolha do algoritmo correto na hora de montar o modelo preditivo do experimento no *Azure Machine Learning*.

Voltando para um exemplo de Regressão Linear simples, pense hipoteticamente que a especificação do imóvel se dá apenas por uma única variável. Acompanhe na Figura 112 como seria um a regressão linear de um *dataset* da UCI sobre os preços de casas em Boston<sup>22</sup>. Neste caso, as duas variáveis utilizadas para o exemplo são LSTAT que significam os status baixo da população em percentual e MEDV que é a mediana do valor de casas ocupadas por seus próprios donos dividido em um fator de 1000 (isso significa que cada valor do eixo Y foi dividido por 1000 para apresentação no *dataset*).

---

<sup>22</sup> Dataset da UCI – Universidade da Califórnia – Irvine: <https://archive.ics.uci.edu/ml/datasets/Housing>

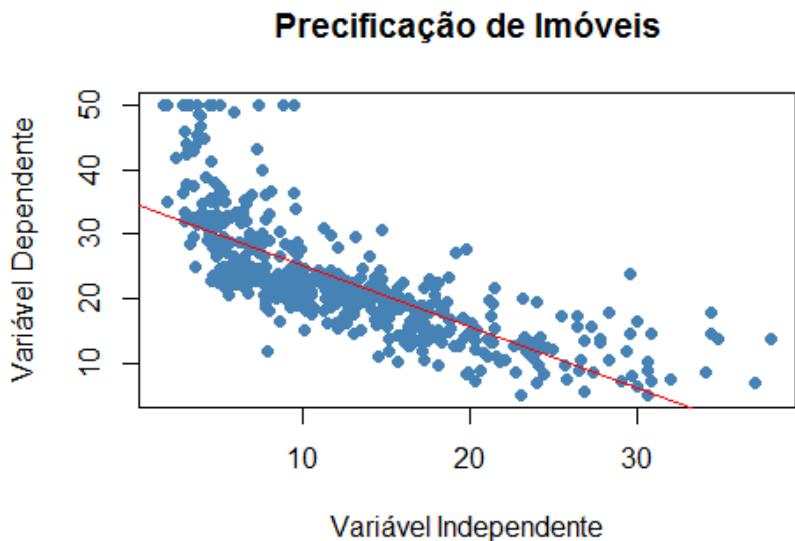


Figura 112 - Precificação de Imóveis

## Regressão Logística

Este modelo de regressão se difere do anterior pois aqui o que se busca é a probabilidade de algum evento acontecer ou não. Mais uma vez as variáveis independentes serão responsáveis por determinar o resultado que terá a variável dependente, que neste caso, a variável dependente será binária, valores discretos, e não mais valores contínuos (infinitos).

Para simplificar esta etapa, o resultado que se busca é um valor normalizado entre 0 e 1, e a função logística ajuda a definir se a classificação binária provável é de 0 ou de 1. Em termos gerais, a busca é por um valor  $0 \leq h\theta(x) \leq 1$ .

A equação padrão para uma regressão logística pode ser representada por esta a seguir:

$$h\theta(x) = g(\theta^T x)$$

Que pode ter variações para:

$$h\theta(x) = g(z)$$

Onde o ( $z$ ) continua sendo o ( $\theta^T x$ ). Chegando na terceira e última variação da equação:

$$h\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Você pode abstrair esta complexidade matemática simplesmente sabendo que o resultado deste algoritmo é uma classificação binária, que terá uma fronteira de decisão separando os dados para a probabilidade de ser zero e de ser um. O resultado desta equação genérica, geralmente é apresentada por uma curva sigmoide, como apresentado na Figura 113.

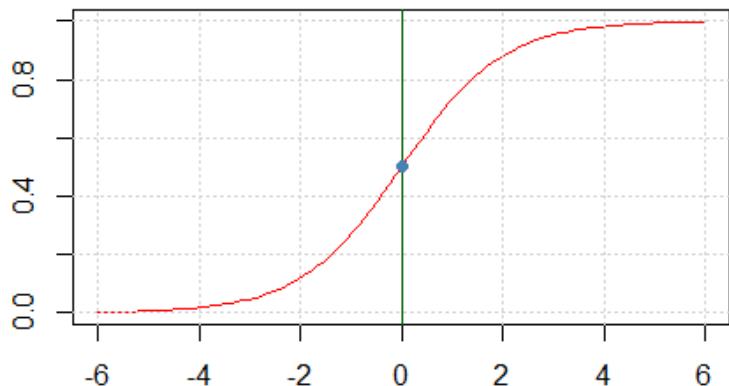


Figura 113 - Função Logística ou Função Sigmoide

Quando se aplica esta função logística em dados reais, é necessário encontrar os coeficientes de regressão para então conhecer a fronteira de decisão que fará a classificação dos dados. Para criar uma representação visual que faça esta classificação como foi feito no exemplo da regressão linear, seriam necessários diversos outros parágrafos de explicação e entendimento profundo dos coeficientes obtidos nas funções logísticas. Como não é o objetivo do livro aprofundar nestas especificações técnicas e matemáticas, é importante focar seu conhecimento em Regressão Logística para criação de classificadores binários baseados em probabilidade.

## Classificação

Algoritmos de aprendizagem de máquina que se enquadram na categoria de Classificação são problemas que buscam identificar a qual classe, também chamada de *label*, uma nova observação se encaixa dentre um conjunto conhecido de possíveis classes. As classes neste cenário são discretas e pertencem a um conjunto finito conhecido.

Algoritmos pertencentes a estas classes são amplamente utilizados na área de Reconhecimento de Padrões e também são conhecidos como classificadores.

Os algoritmos de classificação são problemas de aprendizagem supervisionada. Existem diversas técnicas amplamente utilizadas e o AzureML implementa a maioria deles: Máquinas de Vetor de Suporte (*Support Vector Machines - SVM*), Redes Neurais (*Neural Networks – NN*), Árvores de Decisão, *Naïve Bayes*, entre várias outras.

Problemas de classificação necessitam de informações, elementos rotulados previamente de todas as classes analisadas. Ou seja, se tivermos um problema que identifica três formas diferentes de câncer, teremos a necessidade de possuir uma base rotulada que contenha elementos para as três classes. Outro ponto importante é que a amostragem necessária destas classes deve ser significativa para representar as mesmas.

É comum problemas de múltiplas classes serem quebrados/divididos em problemas de classificação menores, técnica chamada de redução do escopo, tendo assim classificadores mais especializados resultando em um processo menos genérico. Por exemplo, podemos ter um classificador que recebe padrões de entrada oriundos de imagens para identificar tipos de frutas com diversas possibilidades de frutas como classes, ou podemos ter vários classificadores especializados que diferenciam um par de frutas (diversos classificadores binários).

Como dito, existem diversas formas de classificarmos nossos dados. Nas seções seguintes vamos abordar três principais: SVM, redes neurais, e árvores de decisões binárias.

## SVM - Support Vector Machines

O SVM é um classificador binário para aprendizagem supervisionada, isto é, consegue classificar entre apenas duas classes e para seu treino necessitamos de elementos rotulados de ambas as classes. A exceção é quando utilizamos o SVM para detecção de anomalias, quando podemos treiná-lo com apenas uma classe – ver seção mais adiante neste capítulo.

Foi proposto em 1979 pelo russo *Vladimir Vapnik*, e foi considerado um dos maiores acontecimentos da área de aprendizagem de máquina dos últimos 20 anos.

O SVM é baseado no Teorema de Cover<sup>23</sup> que diz que os dados são propensos a serem separáveis linearmente em altas dimensões. Este algoritmo busca encontrar o hiperplano ótimo através da maximização das margens. Veja a explicação a seguir.

Um problema de duas dimensões onde vamos utilizar um classificador linear pode possuir diversas soluções, conforme ilustrado na Figura 114. Chamamos o hiperplano que divide o modelo (no caso de duas dimensões é uma reta) de fronteira de decisão.

---

<sup>23</sup> Cover, T.M. (1965). "Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition". *IEEE Transactions on Electronic Computers*. EC-14: 326–334. doi:10.1109/pec.1965.264137.

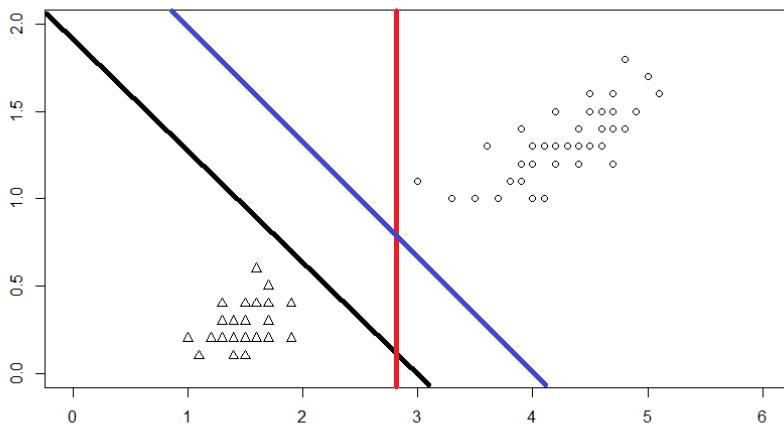


Figura 114 - Exemplo de diferentes fronteiras de decisão

Todas as fronteiras de decisão mostradas separam as duas classes, uma representada por triângulos e outra representada por círculos, mas qual é a melhor?

Note que este é um exemplo de duas dimensões, porém é comum tratar problemas com dimensionalidades muito maiores, isso dependerá do número de características/features utilizadas, o que não seria possível representar graficamente. A fronteira de decisão em um modelo de muitas dimensões é o que chamamos de hiperplano.

A melhor fronteira de decisão, ou hiperplano ótimo, é aquele capaz de generalizar ao máximo a fronteira de decisão. No nosso exemplo simples, seria aproximadamente algo como ilustrado na Figura 115.

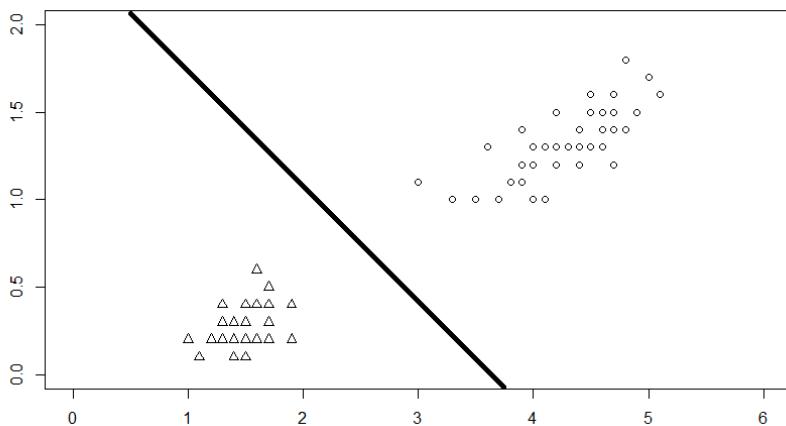


Figura 115 - Representação aproximada da fronteira de decisão ótima

Analisamos diferentes possibilidades de fronteiras de decisão, pois isso facilita o entendimento do comportamento do SVM. O SVM basicamente busca encontrar o hiperplano ótimo que separa dois conjuntos. Isso é alcançado através de um processo conhecido como maximização das margens.

O nome do algoritmo, vetores de suporte, tem origem nos vetores criados no limite dos dados. E através destes vetores que o hiperplano é definido, através de minimização de uma função quadrática. A forma com que isso é calculado está fora do escopo do nosso livro, porém podemos afirmar que existe alto custo computacional envolvido.

A Figura 116 ilustra os vetores de suporte aproximados para a fronteira de decisão encontrada.

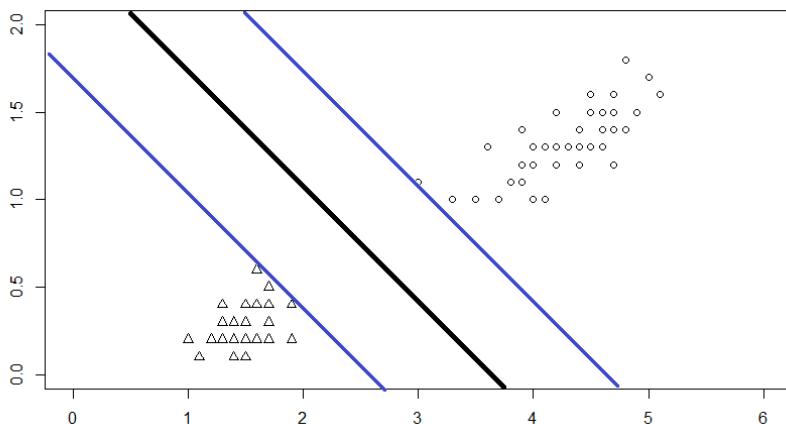


Figura 116 - Vetores de suporte (em azul)

Durante o processo de treino, o SVM pode sofrer com *outliers* e com exemplos rotulados incorretamente. Nestes casos a fronteira de decisão poderia ser gerada incorretamente. Em muitos casos os problemas não são linearmente separáveis.

Mesmo com estes cenários, o SVM ainda assim pode ser aplicado. Para saná-los o SVM implementa o conceito de *soft margin*, onde, de forma simplificada, ele ignora alguns exemplos próximos a fronteira de decisão para encontrar a fronteira de decisão ideal.

Contudo, ainda assim os problemas podem não ser linearmente separáveis. E nestes casos é que o SVM utiliza o que chamamos de *kernel trick*. Esta estratégia eleva a dimensionalidade do problema até um nível onde os dados sejam linearmente separáveis.

Existem diversos *Kernels* utilizados, a saber: Radial, Linear, Gaussiano, Polinomial, Tangente Hiperbólica, entre outros. A Figura 117 ilustra o *Radial Basis Kernel* aplicado a um problema não linearmente separável. Observe que no gráfico a esquerda o problema não é separável por uma reta, duas dimensões. Contudo, no gráfico da direita podemos separar através de um hiperplano após a aplicação da técnica de *kernel trick*, elevado a um espaço de três dimensões.

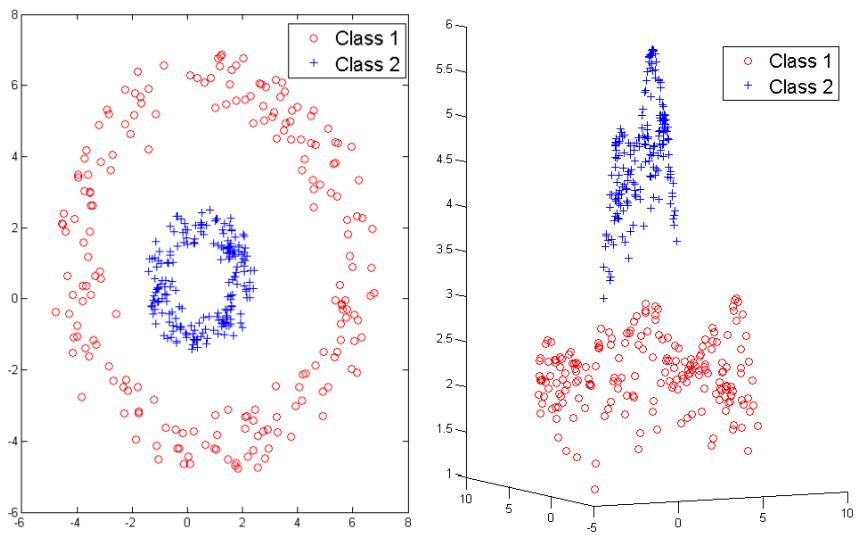


Figura 117 - Aplicação do Kernel trick (duas x três dimensões)

Dado que o classificador esteja treinado, para efetuar a classificação de uma nova observação devemos elevar ela à mesma dimensionalidade, para isso utiliza-se o mesmo *kernel*. E assim podemos classificá-lo bastando ver a sua posição em relação ao hiperplano. Ou seja, apesar do processo de treino ser computacionalmente caro, a classificação é extremamente rápida.

O AzureML não permite customização de *kernel* a se utilizar, mas existem parâmetros importantes que são importantes conhecer.

*Normalize features:* O SVM trabalha melhor com valores normalizados. Esta opção está ativa por padrão. Na maioria dos casos você não vai necessitar mudar.

*Number of iterations:* número de iterações que o algoritmo irá utilizar para localizar a fronteira de decisão. Por padrão é 1. Ao aumentar este número é possível conseguir resultados melhores, maior acurácia, contudo é uma troca de acurácia em relação à velocidade de treinamento.

*Lambda:* peso utilizado na regularização do algoritmo. Para seus primeiros testes mantenha o valor padrão antes de testar com outros. Valores diferentes de zero ajudam a prevenir o *overfitting*, quando o modelo “decora”

o problema e adequa o resultado ao que ele conhece – que são os dados de treino.

## Support Vector Machines para problemas não binários

Como vimos na seção anterior, o SVM consiste em um classificador binário, porém a grande maioria dos problemas reais não possuem apenas duas classes: identificação de caracteres manuscritos; placas de automóveis, tipos de câncer, expressões faciais, etc.

Dentro do AzureML temos um módulo que trata uma das estratégias possíveis para utilizar um classificador binário com múltiplas classes: *Um contra todos*, encontrado no portal como *One-vs-All Multiclass*.

A estratégia *um contra todos* é baseada na criação de N classificadores, onde N é igual ao número de classes. Cada classificador desses é utilizado com uma das classes em questão e a classe secundária composta pelos elementos das outras classes. A classe secundária recebe um rótulo único, veja exemplo adiante.

É importante ressaltar que esta técnica pode ser aplicada para problemas de classificação utilizando outros classificadores binários.

Durante o processo de classificação é atribuído como rótulo o valor referente ao classificador que apresentou o maior grau de confiança no resultado apresentado.

Imagine que temos um problema que busca identificar de forma automática espécies de peixes: salmão, robalo ou atum. Segundo a restrição binária, o SVM não se aplicaria. Para isso utiliza-se a estratégia de *um contra todos*.

Esta técnica envolve criar 3 classificadores.

- Classe 1: salmão | Classe 2: demais observações (robalo, atum)
- Classe 1: robalo | Classe 2: demais observações (salmão, atum)
- Classe 1: atum | Classe 2: demais observações (salmão, robalo)

Para se classificar uma nova observação, aplicamos os valores de entrada aos três modelos e o que responder com o maior grau de confiança, atribuímos como resultado da classificação àquela classe.

Parece complicado? Pois saiba que o AzureML já implementa tudo isso de forma transparente para você, e basta adicionar um único módulo, que não possui parâmetros de configuração.

A Figura 118 mostra a utilização de dois algoritmos, *SVM* e *Multiclass Decision Forest*, para um problema de múltiplas classes. A única diferença é a utilização do módulo *One-vs-All Multiclass*.

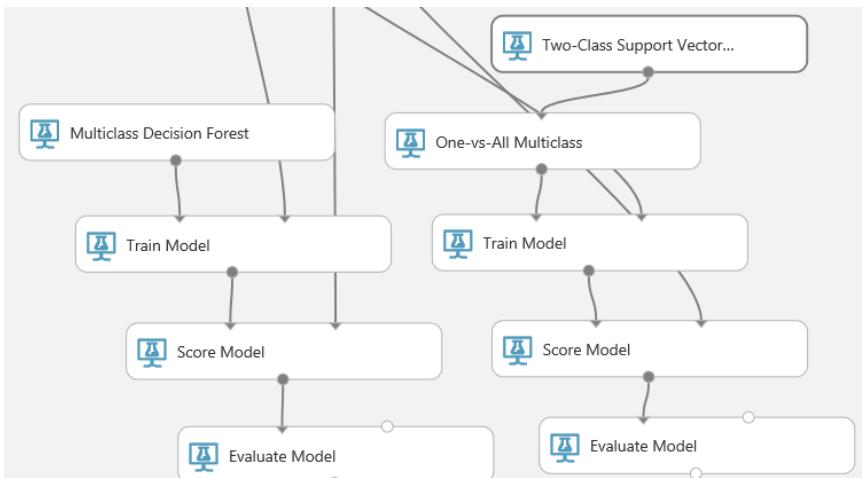


Figura 118 - Utilização do SVM (Two-Class Support Vector Machine) e do módulo One-vs-All Multiclass

O módulo *One-vs-All Multiclass* também pode ser utilizado juntamente a outros algoritmos originalmente desenhados para apenas duas classes.

## Árvores de Decisão, Decision Trees, e Árvores de Decisão Otimizadas, Boosted Decision Trees

O método de árvores de decisão é um modelo preditivo que mapeia as observações segundo conjuntos de regras organizados em um formato de árvore. As árvores de decisão podem atender a cenários de classificação e também regressão, *CART - Classification and Regression Tree*.

Para o *Azure Machine Learning* existem diversas opções de árvores, ou derivados, implementados. Entre eles podemos citar:

- Classificação
  - Two-Class Boosted Decision Tree
  - Two-Class Decision Forest<sup>24</sup>
  - Two-Class Decision Jungle<sup>25</sup>
  - Multiclass Decision Forest<sup>26</sup>
  - Multiclass Decision Jungle<sup>27</sup>
- Regressão (ver mais na seção de modelos de Regressão anteriormente neste capítulo)
- Boosted Decision Tree Regression<sup>28</sup>
- Decision Forest Regression<sup>29</sup>

---

24 Two-Class Decision Forest: <https://msdn.microsoft.com/en-us/library/azure/dn906008.aspx>

25 Two-Class Decision Jungle: <https://msdn.microsoft.com/en-us/library/azure/dn905976.aspx>

26 Multiclass Decision Forest: <https://msdn.microsoft.com/en-us/library/azure/dn906015.aspx>

27 Multiclass Decision Jungle: <https://msdn.microsoft.com/en-us/library/azure/dn905963.aspx>

28 Boosted Decision Tree Regression: <https://msdn.microsoft.com/en-us/library/azure/dn905801.aspx>

29 Decision Forest Regression: <https://msdn.microsoft.com/en-us/library/azure/dn905862.aspx>

Os algoritmos derivados não são cobertos neste livro, porém você pode encontrar maiores informações nas notas de referência nos rodapés das páginas, mas em geral buscam diminuir a convergência pra *overfitting* (decorar) do problema. Em geral utilizando *ensembles*<sup>30</sup> e outras técnicas.

Em uma árvore de decisão tradicional, conhecida como *Decision Tree*, a estrutura do classificador é organizada no formato de uma árvore. Cada nó intermediário, não folha, da árvore corresponde a uma característica e um condicional aplicado a mesma. Os nós folha correspondem às classes efetivamente. Este formato de organização é conhecido como dendrograma e pode ser visualizado de forma gráfica, o que em geral facilita a observação de como a classificação está sendo feita.

A seguir temos um exemplo de um problema com a utilização de uma árvore de decisão para a sua resolução.

Um problema acadêmico clássico de classificação é a construção de um modelo para prever os sobreviventes do naufrágio do *Titanic*. Uma das formas fazer esta predição é através de um modelo baseado em árvore de decisão. Observe que árvore de decisão é uma sugestão para a resolução deste modelo, existe um conjunto de técnicas mais avançadas que podem levar a melhores resultados, e estamos empregando Árvore de Decisão aqui afim de mostrar a sua construção, utilização e visualização apenas. Os dados aqui apresentados são disponibilizados através de um desafio da plataforma Kaggle<sup>31</sup>.

A história conta que durante o naufrágio do *Titanic*, pela pouca quantidade de botes salva-vidas, foi empregado uma política de salvar mulheres e crianças primeiro.

O script em R a seguir mostra como construir um modelo baseado em árvore para resolver este problema utilizando as características de sexo, idade e a classe da pessoa. Caso R não seja de grande familiaridade você pode acompanhar o Capítulo 09 que descrevemos os fundamentos da linguagem,

---

30 [Wikipedia Ensemble Learning:](https://en.wikipedia.org/wiki/Ensemble_learning)  
[https://en.wikipedia.org/wiki/Ensemble\\_learning](https://en.wikipedia.org/wiki/Ensemble_learning)

31 Kaggle Titanic: <https://www.kaggle.com/c/titanic>

contudo os comentários no código já devem facilitar o entendimento. Estamos utilizando R pois o AzureML não disponibiliza a árvore de decisão tradicional e queremos visualizar o seu dendrograma.

Os dados que são disponibilizados para este problema contemplam uma série de características, contudo para a nossa demonstração utilizaremos apenas as apresentadas abaixo:

- Sobrevivência (Survival): 0 não sobreviveu; 1 = sobreviveu
- Classe do passageiro (Pclass): 1 = 1<sup>a</sup> classe; 2 = 2<sup>a</sup> classe; 3 = 3<sup>a</sup> classe
- Sexo (sex)
- Idade (age)

A Figura 119 mostra um subconjunto dos dados nativos fornecidos junto ao problema no Kaggle para as características (*features*) desejadas.

Survived	Pclass	Age	Sex
0	3	22.00	male
1	1	38.00	female
1	3	26.00	female
1	1	35.00	female
0	3	35.00	male
0	3	NA	male
0	1	54.00	male
0	3	2.00	male
1	3	27.00	female
1	2	14.00	female
1	3	4.00	female
1	1	58.00	female
0	3	20.00	male
0	3	39.00	male
0	3	14.00	female
1	2	55.00	female
0	3	2.00	male
1	2	NA	male

Figura 119 - Exemplo contendo os dados dos passageiros do Titanic

O script para a construção de uma árvore utilizando R pode ser visto a seguir:

```
# Instala e carrega as bibliotecas que utilizaremos
install.packages('rpart.plot')
install.packages('rattle')
install.packages('RColorBrewer')
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
```

```
#carrega os dados do csv para a memória em uma variável chamada treino  
treino <- read.csv("train.csv", stringsAsFactors=FALSE)  
# Cria uma coluna descritiva para a classe (apenas para melhorar a visualização do  
dendrograma)  
treino$Sobreviveu <- 'Não Sobreviveu'  
treino$Sobreviveu[treino$Survived == 1] <- "Sobreviveu"  
  
# Treina a árvore  
arvore <- rpart(Sobreviveu ~ Pclass + Sex + Age, data=treino, method="class", control  
= rpart.control(minsplit = 0, minbucket = 0, maxdepth = 10))  
  
# Exibe o dendrograma  
fancyRpartPlot(arvore)
```

Repare que este não é um exemplo completo de *Machine Learning*, visto que não testamos o modelo treinado com uma outra base para testes. Perceba também que não estamos utilizando nenhuma técnica sobre as características, *feature engineering*, o que seria altamente recomendável para um classificador produtivo. Este é um exemplo apenas para ilustrar o algoritmo.

A Figura 120 mostra a árvore encontrada para o conjunto de treino utilizado.

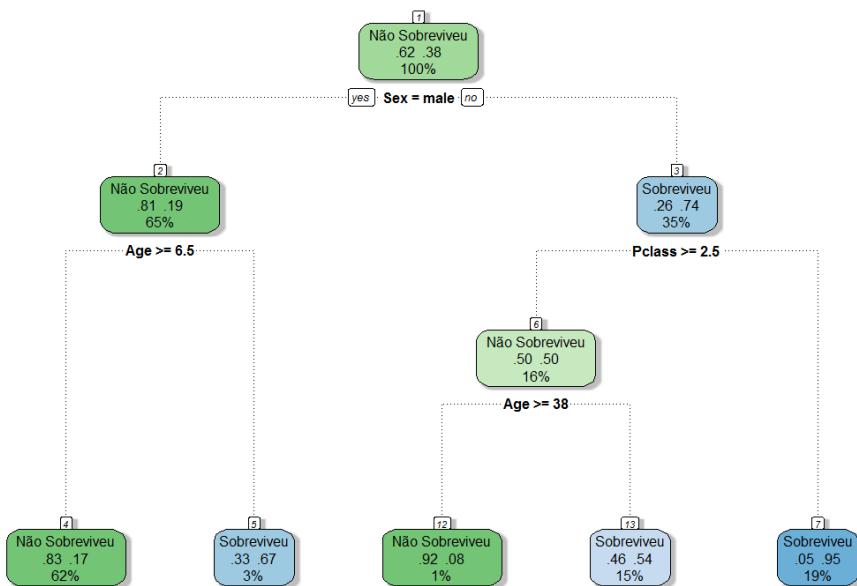


Figura 120 - Dendrograma do problema Titanic

Com a Figura 120 podemos observar a construção que a árvore teve ao se considerar apenas três características: sexo, idade e a classe.

É importante frisar que diversas pessoas no conjunto de dados não possui a idade informada. Uma estratégia caso queira investigar o problema mais a fundo é substituir estes valores faltantes por valores estimados. Uma ideia pode ser utilizar o título da pessoa (*Miss*, *Mr*, etc.), presente no nome, para estimar uma idade.

Como visto, no *Azure Machine Learning* temos à nossa disposição para classificação diversos métodos baseados em árvore, entre eles temos o módulo *Two-Class Boosted Decision Tree* que cria um modelo baseado no algoritmo de mesmo nome.

Uma árvore de decisão aprimorada, *Boosted Decision Tree*, é um método de aprendizagem de máquina supervisionado baseado em um *ensemble*, ou seja, um conjunto de classificadores de árvore de decisão. Neste caso são várias árvores onde a segunda corrige os erros da primeira, a terceira

corrigir os erros da segunda e da primeira e assim por diante. A predição ocorre com a utilização de todas as árvores juntas no *ensemble*.

Este método, se bem configurado, pode levar a resultados interessantes rapidamente, porém é um método que utiliza intensamente os recursos de memória da máquina. Uma das vantagens de se utilizar o AzureML com este método é a escalabilidade e recursos que possui à disposição para treinar seus modelos, tudo de forma transparente.

O algoritmo utilizado para aprimoramento, *boost*, das árvores é o definido em: *Greedy Function Approximation: A Gradient Boosting Machines*<sup>32</sup>. O mesmo está brevemente descrito a seguir.

- Comece com um *ensemble* vazio de classificadores fracos, conhecidos como *weak learners*.
- Para cada exemplo de treino, recupere a saída do *ensemble* como sendo a soma da saída de todos os classificadores no *ensemble*.
- Calcula o gradiente da função de perda.
  - Para classificação é usado *log-loss*, similar a regressão logística.
  - Caso seja para um problema de regressão é usado *squared-loss*, onde o gradiente é a saída atual menos o valor alvo.
- Usa o exemplo para ajustar um classificador fraco usando o gradiente como função alvo.

---

<sup>32</sup> *Greedy Function Approximation: A Gradient Boosting Machines:* <http://www-stat.stanford.edu/~jhf/ftp/trebst.pdf>

- Adiciona o classificador fraco ao *ensemble* com a força, *strength*, igual a taxa de aprendizagem, e se preciso, volta ao passo 2 novamente.

O algoritmo que constrói a árvore seleciona a característica de forma gulosa, conhecido como *greedy algorithm*, sendo a selecionada aquela que mais baixa o *squared-loss* (perda quadrática), em respeito ao gradiente calculado no passo 3. É restrito a um número mínimo de exemplos de treino por folha da árvore em questão. O algoritmo separa, faz o *split*, repetidamente até alcançar o número máximo de nós folha, ou até não ter mais divisões válidas.

Antes do treino efetivo as características são discretizadas, em geral algoritmos de árvore se comportam melhor com características discretas, mas isso não é um pré-requisito. Valores contínuos podem ser discretizados através de uma técnica conhecida como *data binning*, ou *bucketing*, onde valores contínuos próximos podem ser agrupados como sendo o mesmo, antes do treinamento. Um exemplo clássico da utilização desta técnica é a utilizada em histogramas<sup>33</sup>.

Você pode criar estes grupos discretos manualmente, caso seja de seu conhecimento que isso possa melhorar o resultado esperado. A idade, ou o peso podem ser casos clássicos que, em geral, possam ser interessantes combinar/discretizar manualmente antes de aplicar um algoritmo de árvore. Veja na tabela a seguir uma sugestão de como ficaria uma combinação manual de idades. Perceba que a vantagem deste método é que existe a flexibilidade para a da forma que melhor convier, de acordo com o conhecimento sobre o problema tratado.

<i>Idade</i>	<i>Idade_Grupo</i>
1	[1-4]
2	[1-4]
5	[5-9]
10	[10-15]

---

<sup>33</sup> Histogramas: <https://pt.wikipedia.org/wiki/Histograma>

13	[10-15]
20	[16-25]
25	[16-25]
50	[50 ou mais]
78	[50 ou mais]

Essa discretização antes do treino implica em menos possibilidades de valores de limiar, amplamente conhecidos como *thresholds*, candidatos para serem considerados, mesmo em problemas utilizando características com valores contínuos.

Agora que conhecemos a estrutura básica das árvores de decisão, bem como o funcionamento do algoritmo de *ensemble* utilizado na implementação de *Boosted Decision Tree* no AzureML, podemos verificar os parâmetros para a sua configuração. Observe que existem valores padrão e que, em geral, são um bom ponto de partida para os testes com o algoritmo.

*Maximum number of leaves per tree*: Especifica o número máximo de folhas por árvore do *ensemble*. O valor padrão é 20.

*Minimum number of samples per leaf node*: Especifica o número mínimo de exemplos para formar um nó folha. O padrão são 10 exemplos.

*Learning Rate*: Define a taxa de aprendizagem inicial. O valor padrão é 0,2.

*Number of trees constructed*: Configura o número máximo de árvores, *weak learners*, que podem ser criadas durante o processo de treino. O valor padrão é 100.

*Allow unknown categorical levels*: Se verdadeiro, um nível adicional é adicionado para cada coluna categórica. Quaisquer níveis no conjunto de teste que não estiverem no conjunto de treino são mapeados para este nível adicional.

## Redes Neurais

Redes Neurais, também conhecidas como Redes Neurais Artificiais, são técnicas computacionais inspiradas no modo com que os nossos neurônios se comunicam e aprendem baseados na experiência. A escala, logicamente, é diferente. O cérebro de um ser vivo tem na ordem de bilhões de neurônios, enquanto que uma rede neural típica tem na casa de centenas ou milhares de unidades de processamento.

Como dito a origem do pensamento para a construção do modelo de rede neural veio da biologia. A Figura 121<sup>34</sup> ilustra um neurônio de forma simplificada. A comunicação dos neurônios é feita a través de sinapses, onde os impulsos nervosos são transmitidos do corpo do neurônio através de seu axônio até “atingir” os dendritos de outros neurônios e assim por diante. De forma mais direta ainda, podemos dizer que o neurônio recebe um estímulo, processa e retorna uma saída.

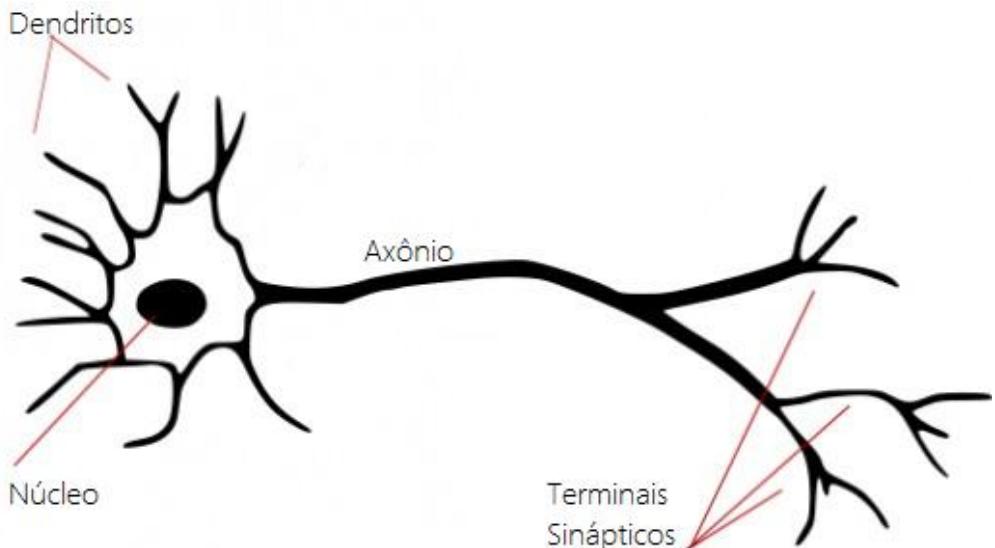


Figura 121 - Neurônio simplificado

<sup>34</sup> Fonte (adaptado): <http://hubpages.com/education/What-is-a-Neuron2#>

Os primeiros trabalhos na área de redes neurais datam dos anos 1940-50 e é uma área que pode gerar bastante discussão e com complexidade elevada. Ao longo da seção buscamos abstrair um pouco da complexidade em ordem de facilitar o entendimento. Para os que buscam um conhecimento avançado deste tópico há uma excelente referência<sup>35</sup> que pode ser consultada para estudo aprofundado.

E por um longo período se resumiu não a uma rede, mas sim a um único “neurônio”, chamado de *Perceptron*, que também está disponível no AzureML<sup>36</sup>. Um *Perceptron* simula o mesmo comportamento de receber um ou mais valores, também chamado de estímulos; processar, conhecido como função de ativação; e retornar uma saída. É considerado a forma mais simplificada de uma Rede Neural Artificial. As entradas são feitas através das suas características/features e são ajustadas com um peso sináptico ajustável, representado pela variável  $w$ , somados e aplicados a uma função de ativação, em geral uma função *sigmoid* (sigmoide semelhante à encontrada na Regressão Logística), mas pode ser outra, e então produzir uma saída, representada pela variável  $y$ , que define a classe. É importante ressaltar que um *Perceptron* apesar de atrativo por ser bastante simples, apenas pode criar fronteiras de decisão em problemas linearmente separáveis. Um *Perceptron* pode ter uma outra entrada opcional, além das originais, conhecida como *bias* e serve para influenciar/deslocar a fronteira de decisão. Os pesos são calculados durante seu treino e são ajustados de acordo com o erro no resultado aplicado à função de ativação. A figura 122 ilustra um *Perceptron* de forma simplificada.

---

35 *Pattern Recognition and Machine Learning*. Christopher Bishop. Springer 2006. ISBN 978-0-387-31073-2

36 Two-Class Averaged Perceptron: <https://msdn.microsoft.com/en-us/library/azure/dn906036.aspx>

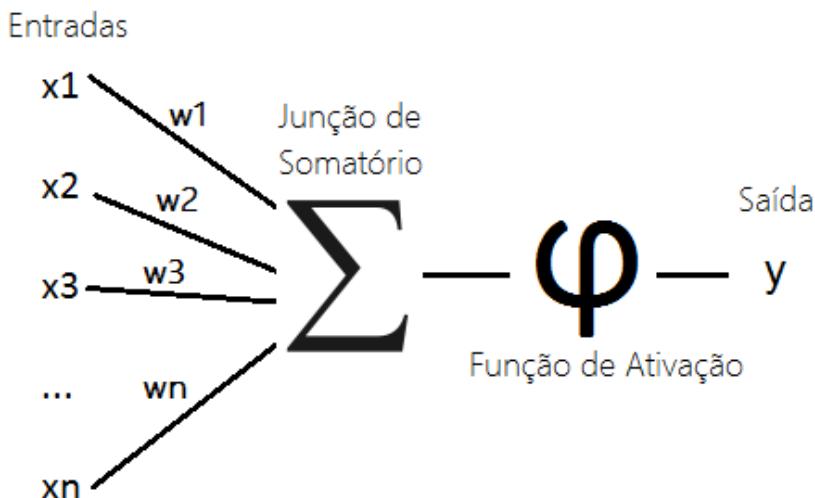


Figura 122 - Perceptron simplificado

Podemos pensar em uma rede neural como sendo um conjunto de diversos *Perceptrons* formando uma verdadeira rede. Esta rede é organizada em camadas (*layers*).

Tipicamente uma rede neural contém uma camada para as características de entrada, recebendo um número de nós iguais ao número de características. Uma ou algumas camadas intermediárias conhecidas como camadas escondidas (do inglês *hidden layers*). E uma camada de saída com o número de nós igual ao número de classes (*labels*) do problema analisado.

Um modelo bastante simples de uma rede neural pode ser observada na Figura 123.

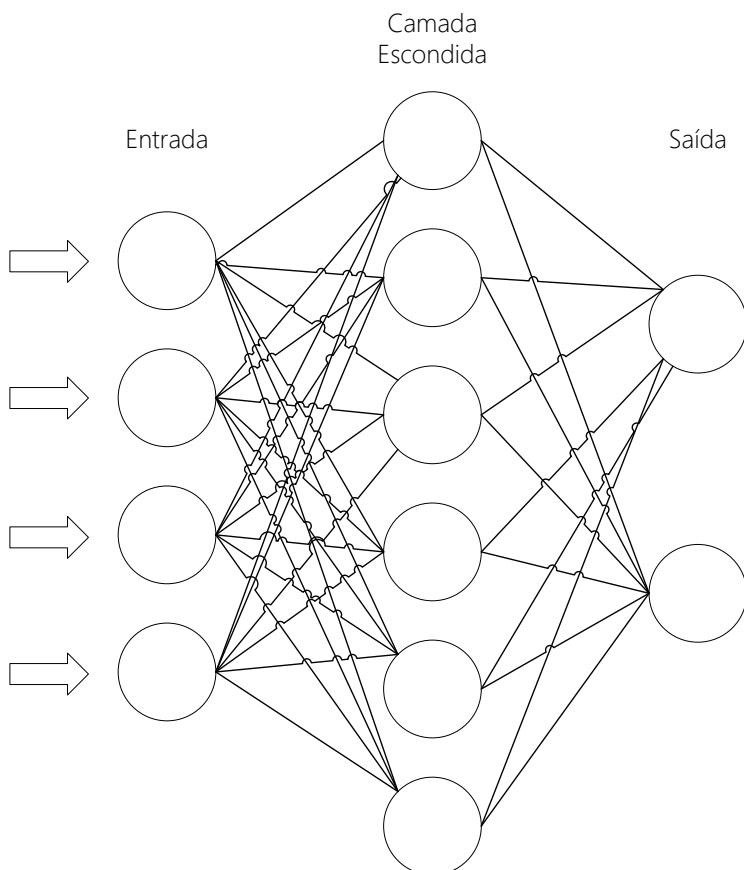


Figura 123 - Modelo simplificado de uma rede neural totalmente conectada

Cada ligação entre os nós é direcionada, seguindo da esquerda para a direita, e possuem pesos, para ilustrar, imagine diversos *Perceptrons* interconectados. Cada nó possui uma função de ativação que determina a saída a ser passada para a camada adiante.

A quantidade de nós na camada escondida, bem como a quantidade de camadas escondidas são determinados pelo cientista de dados e a experimentação de diversas configurações é recomendada.

Os exemplos conhecidos são aplicados ao modelo durante o processo de treino, e a cada aplicação os dados dos pesos são ajustados de acordo com o erro encontrado na camada de saída. Durante muitos anos esse processo não era possível de ser realizado pois não era possível calcular o erro na camada escondida. Isso foi superado com a integração de uma técnica conhecida como *back propagation*. A explicação aprofundada desta técnica está fora do escopo deste livro, porém imagine que com isso foi possível calcular o erro encontrado e reajustar os pesos das conexões.

Durante o processo de treino, todos os elementos do conjunto de treino podem ser aplicados diversas vezes no modelo para ajustar os pesos de acordo com o erro encontrado na camada de saída.

Importante dizer que é possível criar redes neurais sem a camada escondida, mas isso só se aplicaria a problemas linearmente separáveis, o que na maior parte das vezes pode ser solucionado com outras técnicas mais simples ou com treino mais rápido.

Diversas camadas escondidas e nós nas mesmas, levam à construção de fronteiras de decisão mais complexas e poderosas. Contudo, isso acarreta em um tempo maior de treino e computação necessária dado o grande número de pesos que necessitam serem calculados. Em geral é uma boa prática começar com apenas uma camada escondida e não mais que o dobro de nós de entrada na camada escondida. No caso específico do AzureML o valor padrão para esta configuração é de 100 nós, o que também pode ser um bom ponto de partida. Contudo, isso pode variar muito caso a caso, principalmente depois do advento do *deep learning*, técnica para utilizar redes neurais com várias camadas escondidas.

Como mencionamos, cada nó da camada de saída corresponde a uma classe do problema. Na Figura 15 temos a ilustração de um problema de duas classes. Se um novo exemplo é utilizado como entrada para o modelo, o nó da camada de saída com maior valor é usado para atribuir o rotulo da classe.

Imagine que o primeiro nó é a classe A e o segundo nó é a classe B. Ao apresentar uma nova observação o valor de saída do primeiro nó é 0,96 e o do segundo nó é de 0,04. Neste caso atribuímos a classe A para esta observação.

No *Azure Machine Learning* temos três módulos para o uso de Redes Neurais em três cenários diferentes, dois para classificação, duas classes e multi classes; e um para regressão, que teve uma breve explicação do tipo de algoritmo em uma seção anterior.

- Multiclass Neural Network<sup>37</sup>
- Two-Class Neural Network<sup>38</sup>
- Neural Network Regression<sup>39</sup>

A seguir temos a definição dos parâmetros mais importantes ao se utilizar redes neurais na plataforma.

O primeiro parâmetro importante é o *Hidden layer specification*, que define as configurações relacionadas à camada escondida. Existem duas opções: *fully-connected case* e *custom definition script*.

A primeira opção, *fully-connected case*, define o seguinte comportamento:

- A estrutura padrão tem uma camada escondida. Ou seja, não é para modelos de aprendizagem profunda, conhecidas como *deep learning*.
- A camada de saída é totalmente conectada na camada escondida e a camada escondida é totalmente conectada na camada de entrada. Similar ao apresentado na Figura 15.
- O número de nós da camada de entrada é determinado pelo número de características, features, do conjunto de treino.

---

37 Multiclass Neural Network: <https://msdn.microsoft.com/en-us/library/azure/Dn906030.aspx>

38 Two-Class Neural Network: <https://msdn.microsoft.com/en-us/library/azure/dn905947.aspx>

39 Neural Network Regression: <https://msdn.microsoft.com/en-us/library/azure/dn905924.aspx>

- O número de nós na camada escondida é definido pelo usuário através do parâmetro *Number of hidden nodes*. Por padrão este valor é 100.
- O número na camada de saída depende do número de classes do problema.

Já a segunda opção, *custom definition script*, permite bastante flexibilidade pois através deste script você pode criar uma arquitetura de rede customizada. A linguagem utilizada é a Net#. A especificação completa desta linguagem está fora do escopo deste livro, mas existe um guia bastante interessante disponível<sup>40</sup>, além da introdução feita pelo Alexey Kamenev do time de produto do AzureML<sup>41</sup>.

Um exemplo da linguagem Net# para uma rede simples pode ser visto a seguir.

```
input Dados auto;  
  
hidden Escondida [100] from Dados all;  
  
output Saida [10] sigmoid from Escondida all;
```

A utilização da palavra “auto”, na primeira linha do script, faz com que a rede neural se adapte para ter na camada de entrada o número de nós equivalente à quantidade de características recebida pelo módulo no AzureML, na prática é a quantidade recebida pelo módulo *Train Model*.

A segunda linha é referente à camada escondida, que possui apenas uma neste caso, onde demos o nome de *Escondida* com 100 nós. Esta camada

---

40 *Guide to Net# neural network specification language for Azure Machine Learning:* <https://azure.microsoft.com/en-us/documentation/articles/machine-learning-azure-ml-netsharp-reference-guide/>

41 *Neural Nets in Azure ML – Introduction to Net#:* <https://blogs.technet.microsoft.com/machinelearning/2015/02/16/neural-nets-in-azure-ml-introduction-to-net/>

é totalmente conectada na camada de entrada, especificado pela sintaxe “from Dados”, onde Dados foi o nome dado para a camada de entrada. Ao não se especificar a função de ativação, a função *sigmoid*<sup>42</sup> é usada por padrão. É possível a utilização de outras também, como por exemplo a *softmax*<sup>43</sup>.

A terceira linha define a camada de saída *output* chamada *Saida* com 10 nós, ou seja, é um modelo para resolver um problema de dez classes. A palavra-chave *sigmoid* especifica a função de ativação utilizada na camada de saída.

A Figura 124 ilustra o exemplo mostrado.

## Properties Project

### ▲ Multiclass Neural Network

Create trainer mode

Single Parameter

Hidden layer specification

Custom definition script

Neural network definition

```
1 input Dados auto;
2 hidden Escondida [100] from Dados all;
3 output Saida [10] sigmoid from Escondida all;
```



Figura 124 - Configuração de script personalizado em Net#

Voltando aos parâmetros dos módulos de redes neurais, temos:

- *Learning rate*: Define o tamanho do passo a cada iteração da rede neural antes da correção baseado no erro. Valores maiores podem convergir mais rápido, significando um treino

---

42 [https://en.wikipedia.org/wiki/Sigmoid\\_function](https://en.wikipedia.org/wiki/Sigmoid_function)

43 [https://en.wikipedia.org/wiki/Softmax\\_function](https://en.wikipedia.org/wiki/Softmax_function)

mais rápido, porém pode ser que não encontre o mínimo local adequado. Pode ser entendido como um ajuste dos pesos. Em geral, comece com valores pequenos e altere caso não pareça adequado. O valor padrão é 0,1.

- *Number of learning interactions*: especifica o número máximo de vezes que o algoritmo irá processar os casos de entrada.
- *The initial learning weights diameter*: especifica o peso dos nós no início do processo de treino. O valor padrão é 0,1.
- *The momentum*: esta propriedade define o valor aplicado durante o treino aos nós de iterações anteriores.
- *The type of normalizer*: Especifica o tipo de algoritmo de normalização que será utilizado para os valores das características aplicados à rede neural.
  - *Binning normalizer*: Cria grupos, *bins*, de tamanhos iguais, e então normaliza cada valor em cada grupo para ser dividido pelo total de grupos.
  - *Gaussian normalizer*: Re-escalas os valores de cada característica para terem média 0 e variância 1.
  - *Min-max normalizer*: Re-escalas todas as características entre 0 e 1. Valor mínimo é tratado como zero, e o valor máximo como 1. O restante dos valores é alterado proporcionalmente. Este é o valor padrão e em geral apresenta bons resultados.
  - *Do not normalize*: não é executada nenhuma normalização. Utilize quando os dados de entrada já estiverem normalizados.
- *Shuffle examples*: opção para embaralhar os elementos de treino entre cada iteração. Essa caixa de seleção está marcada por padrão e recomenda-se que se mantenha assim.

- *Random seed number*: semente para os cálculos dos valores pseudoaleatórios, e assim podendo ser utilizado para ter reproducibilidade entre as execuções.
- *Allow unknown categorical levels*: habilita a criação de um grupo de elementos desconhecidos, *unknown*, para os valores nos conjuntos de treino e validação. Se este valor for desmarcado, o modelo poderá aceitar somente valores contidos nos dados de treino. Caso marcado o modelo pode ser menos preciso para valores conhecidos, mas pode ter melhores previsões para valores novos.

## Cluster K-Means

Seguindo com as explicações e aplicações dos algoritmos, uma das formas que existe para se trabalhar com aprendizado de máquinas é o aprendizado não supervisionado. Diferente do aprendizado supervisionado onde você informa ao computador, na verdade ao modelo preditivo, as classes que ele deve procurar e aprender, no aprendizado não supervisionado não é explicitamente informado ao modelo o que exatamente ele deve procurar nos dados para encontrar o padrão. Para resolver este problema, uma alternativa é recorrer à agrupadores lógicos de segmentação para encontrar similaridade entre os dados da amostra e com isso, definir um padrão e assumir que este padrão encontrado é o que estamos tentando ensinar o computador, que por sua vez, vai aprender a encontrar esse padrão sempre quando for solicitado.

Depois de descoberto o padrão, qualquer item novo que tenha uma similaridade com aquele segmento, aglomerado de dados, ou no inglês Cluster, pode ser inferido que ele “faz parte daquilo”.

Nesta seção será explicado como funciona o algoritmo de cluster conhecido como K-Means, ou K-Média, e as fases necessárias para ter sucesso na aplicação deste modelo.

Apesar do AzureML já possuir o módulo de Cluster K-Means implementado, podemos imaginar que este algoritmo não está modularizado na plataforma e que precisamos utilizá-lo na resolução de um problema. Neste caso, como é possível executar módulos em R ou Python dentro da plataforma, existe uma forma de contornar esta situação hipotética. O Capítulo 9 apresenta uma breve introdução à Linguagem R e como podemos interagir com o AzureML e R, e o Capítulo 10 apresenta uma forma de como implementar o Cluster K-Means no AzureML e R, e depois, consumir os resultados no Power BI.

## AGRUPAMENTO DOS DADOS

Para um entendimento mais simples, depois evoluindo para um experimento mais complexo, o K-Means faz o agrupamento de dados mais

próximos ao centroide existente. A quantidade de centroides pode ser descoberta através de técnicas como o *Elbow Method* ou então sabendo quantos grupos serão encontrados. O  $K$  de *K-Means* é a quantidade de centroides que os dados serão agrupados, e o “Means” remete à forma como o algoritmo calcula a posição do centroide e quais pontos fazem parte daquele grupo. A média da distância entre um ponto e os centroides são calculados à cada iteração, e caso algum valor dos dados estejam mais próximo a outro centroide após a iteração do algoritmo, o ponto muda de centroide.

Para facilitar o entendimento, pense em um conjunto de dados com algumas amostras dispostas nos eixos X e Y, como a Figura 125. Faça um exercício mental e agrupe estes dados baseado em suas similaridades.

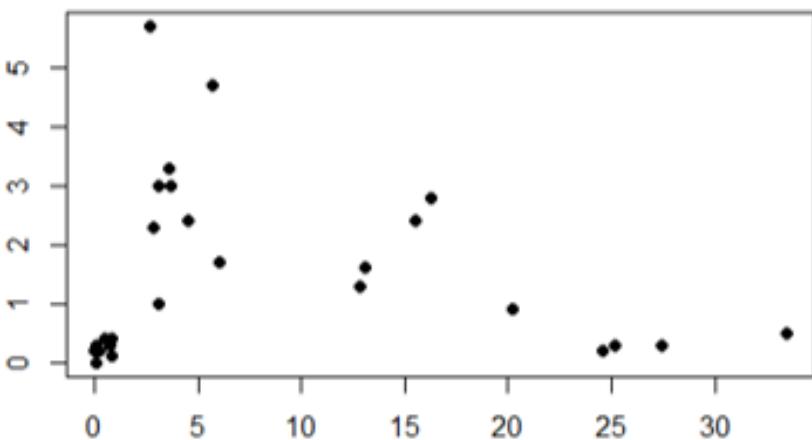


Figura 125 - Dados plotados sem grupos

Descobrir a similaridade simplesmente olhando para o gráfico de pontos como a Figura 125 é difícil e pode muitas vezes ter sido agrupado de formas diferentes. Estas separações diferentes não estão erradas, estão diferentes. O algoritmo *K-Means* nos ajuda a ter um entendimento mais apurado destes agrupadores.

Voltando ao exercício mental, é possível observar este gráfico e ver a separação em alguns grupos. Cada um de nós que olhar o gráfico pode tentar criar um número diferente de agrupadores, também chamados de *clusters*. Até mesmo quando dois de nós pensarmos na mesma quantidade de *clusters*, pode-se pensar em agrupamentos de formas diferentes. Por exemplo, alguns podem ver a separação com apenas 2 clusters, e a separação poderia ser feita como na Figura 126, 127 ou 128. Ambas figuras não estão erradas, estão diferentes.

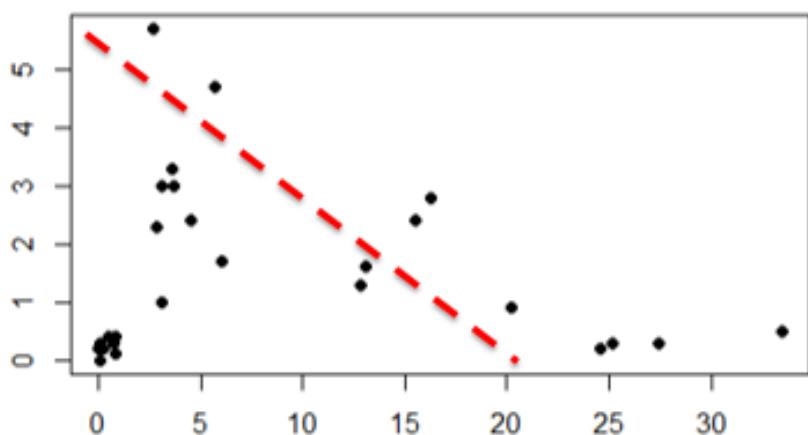


Figura 126 - Separação de 2 grupos - forma 1

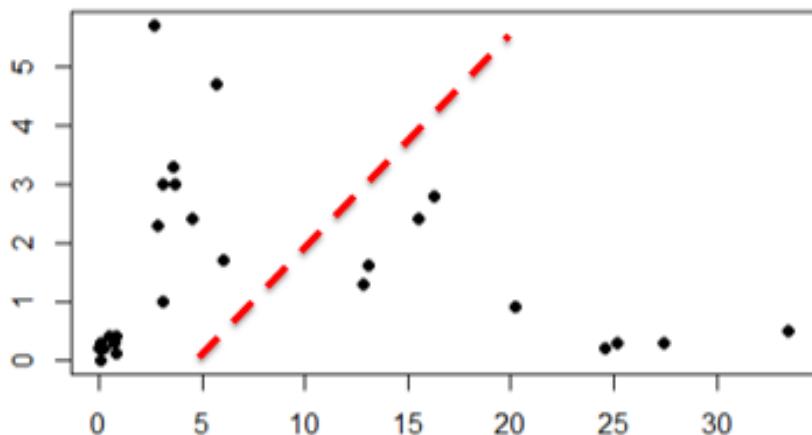


Figura 127 - Separação de 2 grupos - forma 2

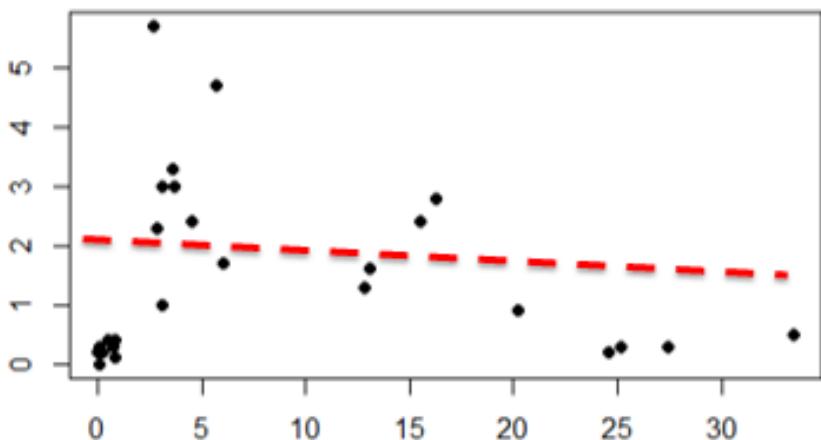


Figura 128 - Separação de 2 grupos - forma 3

Qual é o certo? Todos estão certos! Isso pode acontecer de acordo com a interpretação de cada um dos observadores que encontraram apenas 2 grupos nestes dados. Outros cientistas de dados podem encontrar 3 grupos,

e não apenas dois, podendo chegar a definições como nas Figuras 129, 130 e 131:

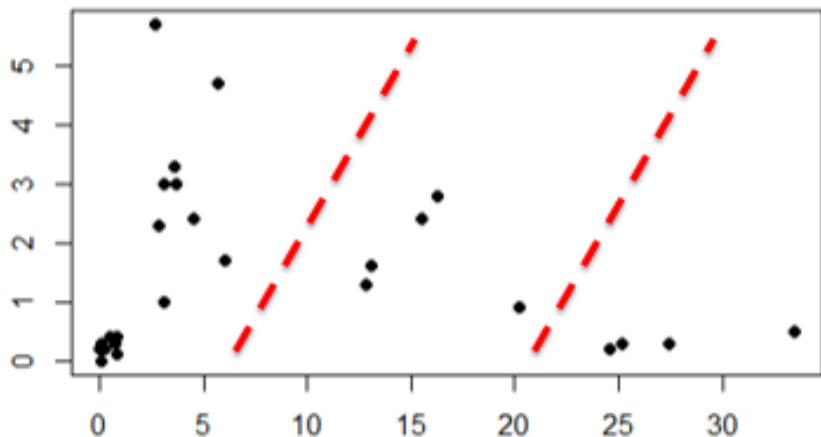


Figura 129 - Separação de 3 grupos - forma 1

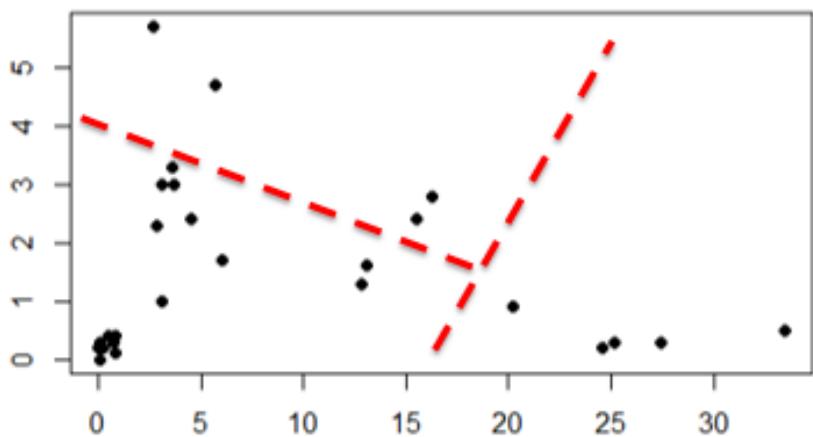


Figura 130 - Separação de 3 grupos - forma 2

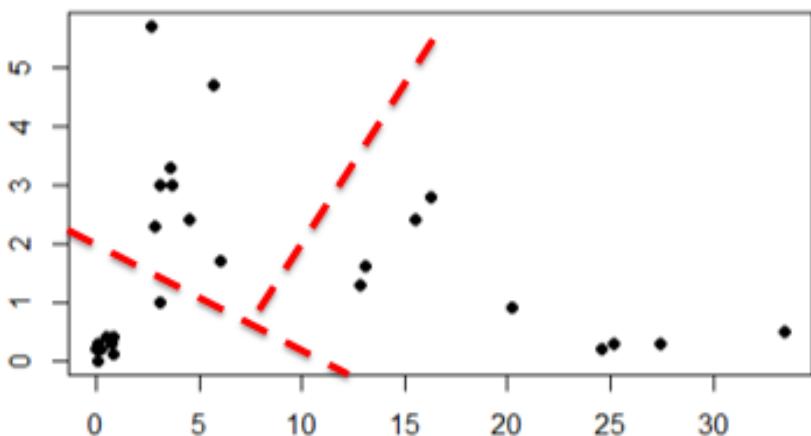


Figura 131 - Separação de 3 grupos - forma 3

E agora com três grupos, qual dos gráficos é o certo? O certo é com 2 grupos ou com 3 grupos?

Mais uma vez isso é difícil de responder baseado apenas em nossa experiência empírica, todos os 6 gráficos estão corretos de acordo com a visão de cada observador.

Para ajudar a responder esta questão, existem alguns métodos usados e bem aceitos no meio científico. Um destes métodos, conforme comentado, é o *Elbow Method*, que terá comentado em uma seção adiante.

## ENTENDENDO COMO FUNCIONA O ALGORITMO

Para entender o funcionamento deste algoritmo mais uma vez começamos de forma simples e depois vamos para um exemplo mais complexo. Com o *dataset* de exemplo que montamos, precisamos separar em 2 grupos e entender os passos que o algoritmo *K-Means* faz com os dados para convergir em um resultado.

Neste caso o  $K$  será igual a 2, criando os 2 *clusters* que estamos buscando. Uma das formas de iniciar o processo é o algoritmo inserir o  $K$

pontos, também chamados de centroides, aleatórios iniciais. Pode ser qualquer lugar do plano, para em seguida começar as iterações e encontrar os resultados.

Veja dois pontos aleatórios criados no gráfico existente na Figura 132, e uma linha tracejada que é calculada aproximadamente na metade da distância dos pontos de dados e dos centroides Vermelho e Azul. Neste passo os itens de dados que estão representados acima da linha tracejada imaginária fazem parte do grupo vermelho e os de baixo da linha fazem parte do grupo azul.

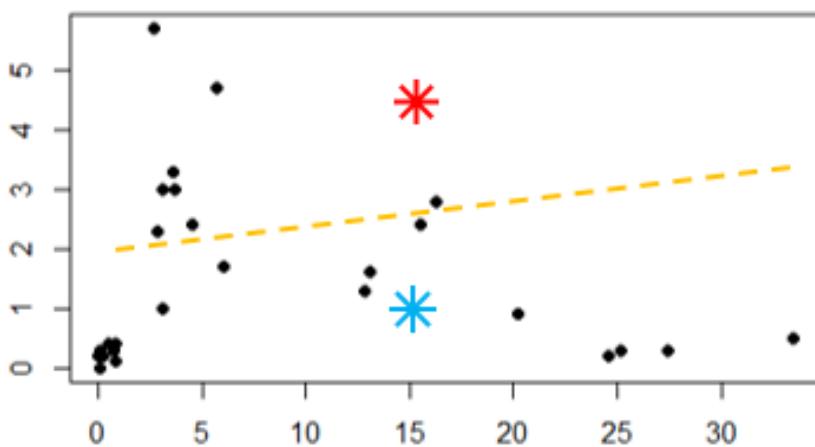


Figura 132 - 2 K aleatórios no gráfico

A primeira iteração do algoritmo é calcular a distância média de todos os pontos que estão próximos ao centroide, e então ajustar a posição do centroide para o novo ponto que foi calculado. Que é a distância média de todos os pontos que se ligaram àquele centroide. Essa mudança de posição do centroide pode alterar os itens que fazem parte daquele grupo. Veja esta mudança nas Figuras 133 e 134.

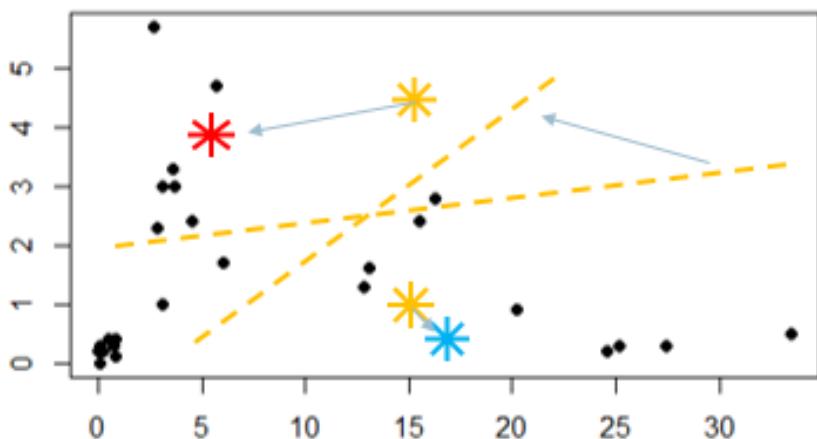


Figura 133 - Movimentação dos centroides para os novos pontos

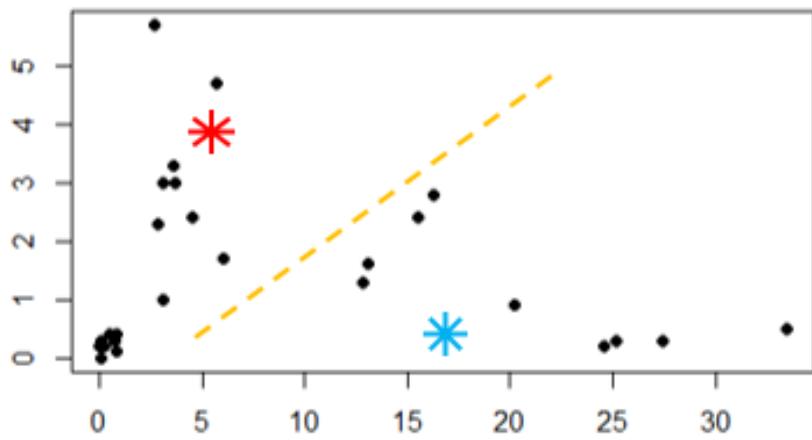


Figura 134 - Novos pontos de centroides

Reparam que após a iteração do cálculo da média, alguns pontos mudaram de centroide. Acompanhe na Figura 135 que os pontos que estão marcados em verde bandeira no lado esquerdo do gráfico passaram inicialmente do centroide azul agora para o vermelho, e o que está sinalizado

em verde petróleo, na marca da direita, passou do centroide inicial vermelho para o azul. Essa iteração de cálculo da média da distância dos pontos até o centroide ocorre em *loop* até que nenhum ponto mude de centroide, isso acontece quando os centroides param de se mover porque já estão na posição central da distância entre os pontos.

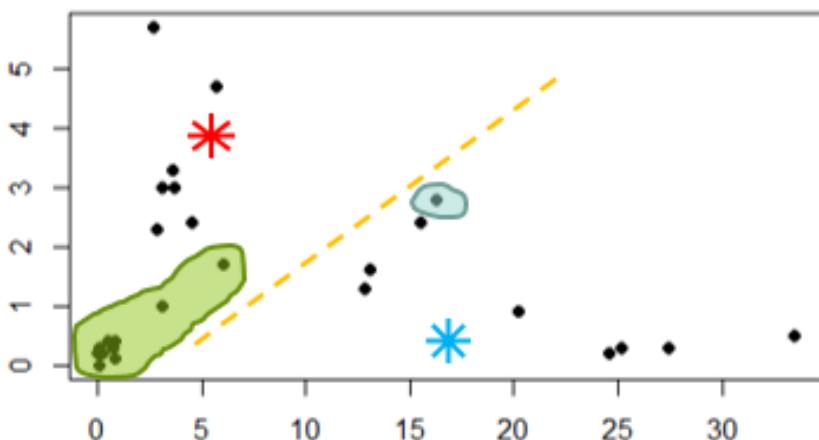


Figura 135 - Pontos mudando de centroide

Mais uma vez o algoritmo faz o cálculo das distâncias médias dos pontos que estão naquele grupo, e então adequa a posição do centroide. Acompanhe nas Figuras 136 e 137 esta nova iteração.

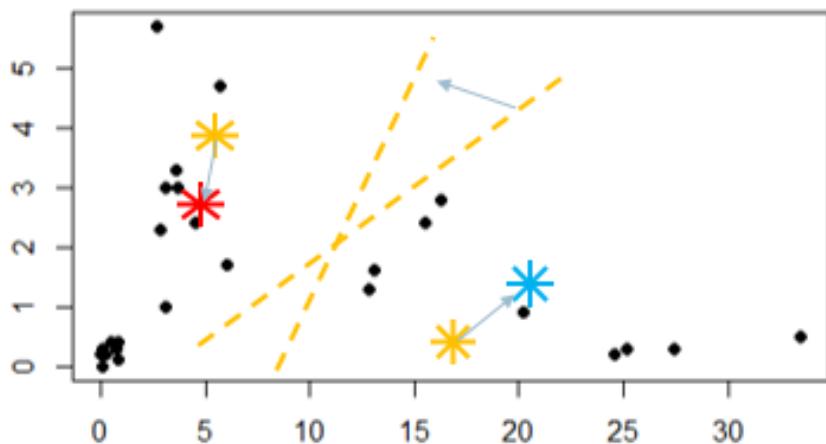


Figura 136 - Movimento dos centroides para os novos pontos

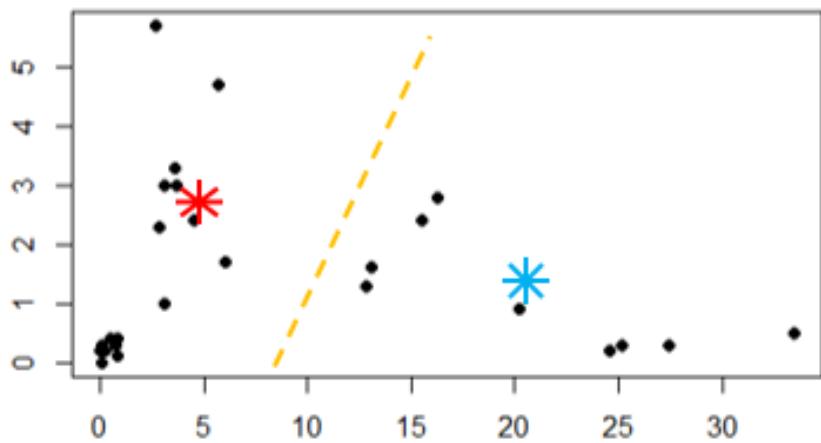


Figura 137 - Nova posição dos centroides

Repare que entre a iteração anterior e esta não ouve mudança de pontos do gráfico entre os centroides, a única alteração foi a posição do centroide. Como não existem mais mudanças nos pontos e centroides, qualquer iteração daqui para a frente o centro não mudará, fazendo com que o algoritmo *K-Means* pare sua execução chegando ao resultado satisfatório.

Acompanhe a Figura 138 como ficaram os pontos do gráfico separados em dois grupos.

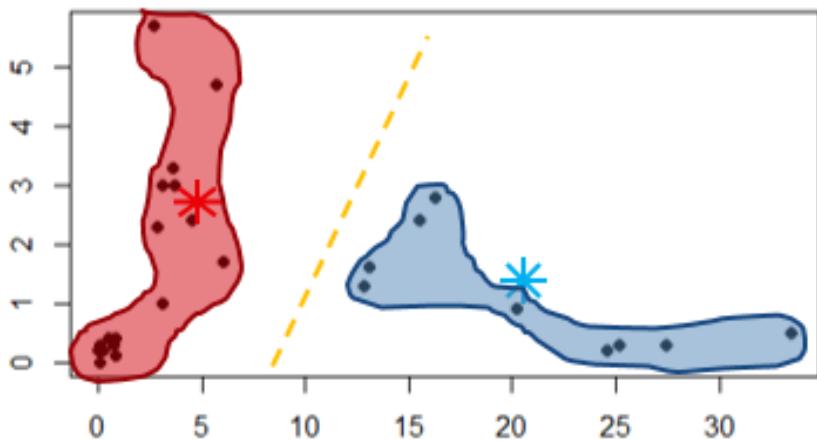


Figura 138 - Dados separados em dois grupos

## ESCOLHENDO A QUANTIDADE DE K (CLUSTERS) NO ALGORITMO

Separar os dados em apenas dois grupos pode não ser o ideal, caso não tenhamos certeza que aquela quantidade de  $K$ s seja o ideal. Para resolver este problema, o *Elbow Method* é uma das formas científicas amplamente usadas. Ele tem esse nome por se parecer com o formato de um “braço” e nós sempre procurarmos o “cotovelo” para definir que este é o número aceitável de  $K$  (*clusters*) a serem criados com base nos dados da amostra.

Este algoritmo faz um laço de repetição e incrementa a quantidade de clusters a partir de 1 e analisa o resultado a cada incremento. Quando o benefício deixar de ser relevante, definido pelo salto entre uma quantidade de cluster e o número seguinte, ele entra em um estado conhecido como platô, no qual a distância da diferença é quase insignificante. É neste momento que

se entende que o algoritmo é relevante com aquela quantidade de K e então ele deve ser usado para segmentar os dados do gráfico.

Depois de executar o código do algoritmo do *Elbow Method* sobre os dados do gráfico plotado nos exemplos anteriores desta seção, descobre-se que um bom número de K para segmentar estes dados de forma satisfatória é de 4 grupos e não apenas 2. Acompanhe a Figura 139 a representação do *Elbow Method* para este conjunto de dados.

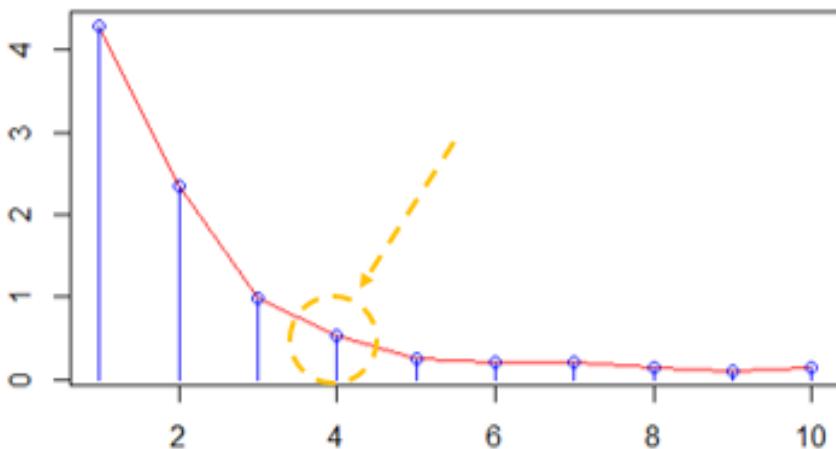


Figura 139 - Elbow Method com os dados do dataset

Agora é conhecida a quantidade correta de agrupadores. Ao executar novamente o algoritmo de *K-Means* com  $k = 4$ , é possível ver a transformação acontecendo como representado na Figuras 140, 141 e 142.

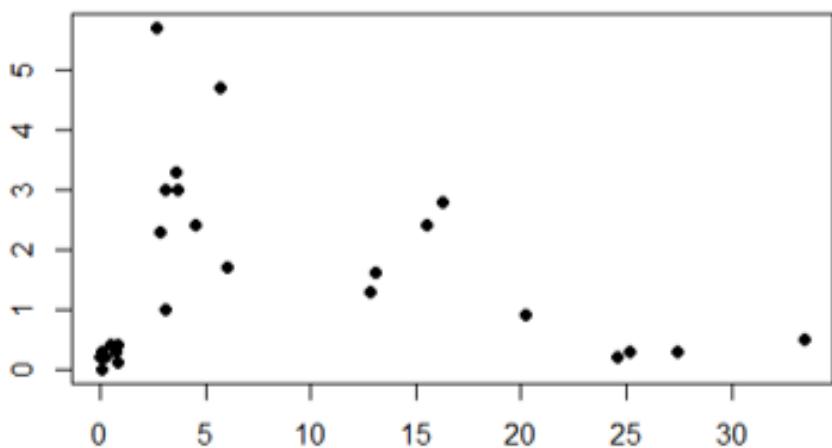


Figura 140 - Dados sem as separações por grupos

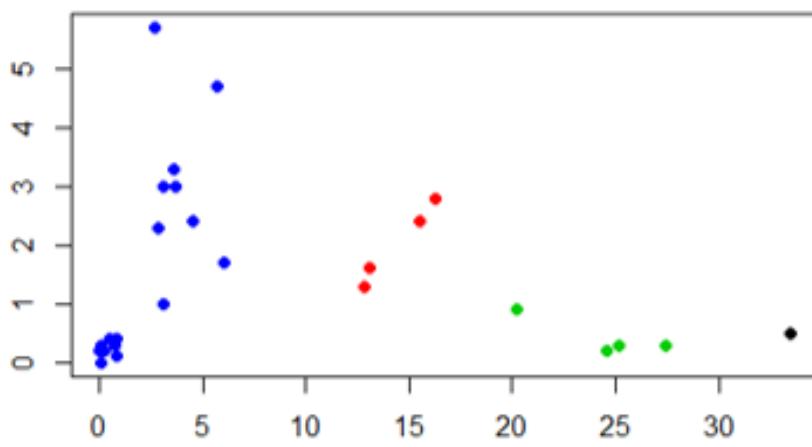


Figura 141 - Dados já com as separações em 4 grupos

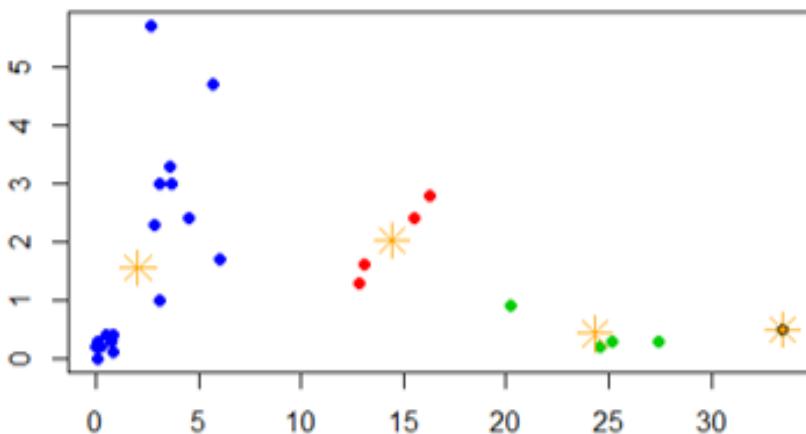


Figura 142 - Dados separados em grupos, com as posições finais de seus centroides

Com estes poucos dados é até imaginável uma forma, ou mais, de fazer a separação dos grupos. Porém, quando a complexidade de dados é maior, fazer a separação dos dados mentalmente se torna cada vez mais difícil. Veja o exemplo de como isso acontece com um *dataset* contendo 60.000 pontos plotados em um gráfico.

Mais uma vez recorremos à uma base de dados da UCI – Universidade da Califórnia, Irvine. Desta vez uma base de atividades esportivas diárias<sup>44</sup>, com uma amostra significativa de dados. Este *dataset* possui a separação de 19 atividades físicas, onde cada atividade foi realizada por 8 pessoas, sendo homens e mulheres de 20 a 30 anos. Não se sabe quem é quem na amostra.

Os dispositivos de coleta foram colocados nos membros superiores, membros inferiores e abdômen. Cada dispositivo possui três sensores, sendo o giroscópio, o acelerômetro e o magnetômetro. Para este exemplo, serão coletados apenas os dispositivos de ambas pernas, ignorando os resultados obtidos nos dispositivos que estão nos membros superiores e no abdômen.

---

<sup>44</sup> UCI – Universidade da Califórnia, Irvine – Atividades esportivas diárias:  
<http://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities>

Neste caso, como é conhecido que são 8 indivíduos que geraram os dados para análise, não é necessário rodar o *Elbow Method* para descobrir quantos clusters devem ser criados. Conhecendo os dados, serão criados 8 grupos, sendo um para cada indivíduo.

A Figura 143 é uma plotagem bidimensional de apenas dados das pernas direitas dos oito indivíduos, coletados dos sensores de acelerômetro nos eixos X e Y. A atividade física escolhida foi *Jogar Basquete*.

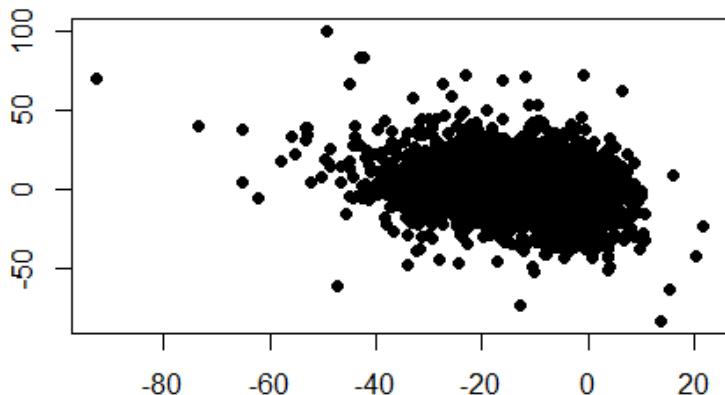


Figura 143 - Dados sem os centroides

Analizar e pensar em oito grupos apenas “mentalmente” é uma tarefa difícil quando o conjunto de dados é mais significante, como este apresentado agora. Esta dificuldade é uma limitação humana, e o algoritmo *K-Means* resolve facilmente. Acompanhe na Figura 144 os pontos centrais de cada grupo.

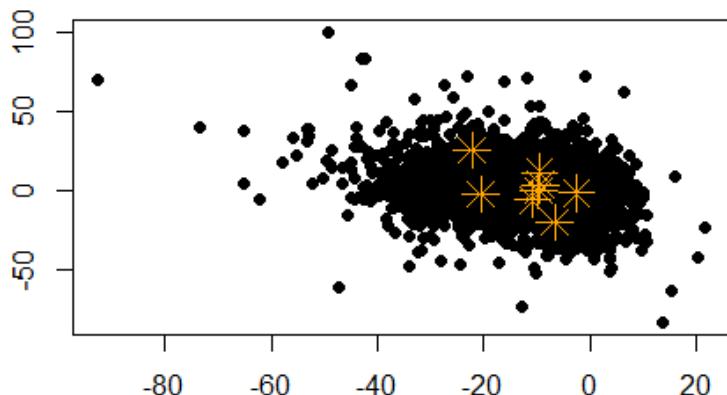


Figura 144 - Dados com oito centroides

Observar apenas os pontos centrais de cada grupo pode continuar dificultando o entendimento visual das separações, pois não sabemos exatamente qual centro recebe cada ponto. Porém, ao colocar uma cor diferente para cada grupo, a distribuição fica bem mais simples de identificar visualmente. Acompanhe como é esta separação na Figura 145 e em seguida na Figura 146 com os centros apresentados.

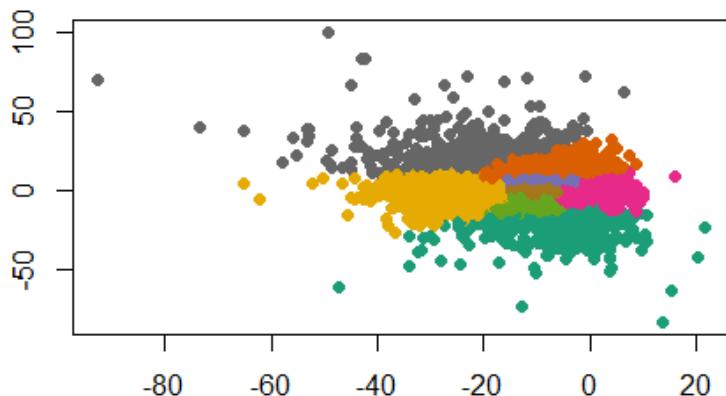


Figura 145 - Dados com cada grupo separado por cor

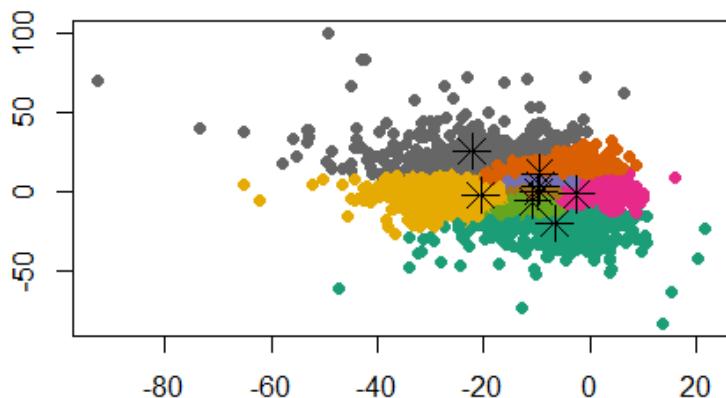


Figura 146 - Dados com cada grupo separado por cor e seu centroide

## Detecção de Anomalias

Detecção de anomalias, também conhecido como detecção de *outliers*, é o grupo de algoritmos/técnicas para encontrar eventos ou observações que não condizem com o padrão regular dos dados.

Podemos entender um *outlier* como sendo um valor atípico, fora do padrão, uma anomalia. Em geral *outliers* podem ser originados de ruídos na fonte de dados utilizada.

Existem diversos problemas em que as técnicas de detecção de anomalias podem auxiliar: detecção de fraudes, detecção de invasões, sistemas de monitoramento de saúde (de sistemas, pessoas, etc.), entre outros. Até mesmo para limpeza de fontes de dados para experimentos de aprendizagem supervisionada, onde a remoção de *outliers* pode aumentar de forma significativa a qualidade, e com isso trabalhamos com melhores resultados, do modelo treinado.

Existem diversas técnicas para detecção de anomalias, entre elas podemos citar: PCA, que é a detecção baseada em análise do componente principal, encontrada amplamente na literatura com seu nome em inglês, *Principal Component Analysis*; *Support Vector Machine* de uma classe, encontrado na plataforma como *One class SVM*; técnicas baseadas em densidade, como o k-NN, etc.; detecção baseada em lógica fuzzy; regras de associação; técnicas baseadas em subespaço e correlação para dados com alta dimensionalidade.

O AzureML implementa dois módulos para detecção de anomalias: SVM de uma classe, encontrado como *One class SVM* e o baseado em PCA, encontrado como *Principal Component Analysis*. É possível expandir esta capacidade através de scripts em R. O processo de como utilizar uma biblioteca customizada do R no AzureML pode ser lido no Capítulo 9.

## Support Vector Machine de Uma Classe

A estrutura e o fundamento do algoritmo são os mesmos já apresentados na seção sobre SVM mais acima neste capítulo, contudo, no âmbito atual ele pode agir como um classificador para detecção de *outliers*. Há também um terceiro cenário – fora do escopo deste livro – onde o SVM pode atuar como um modelo de regressão.

O SVM, como dito, é essencialmente um classificador binário. Para classificadores binários operarem satisfatoriamente demos possuir base de

dados significativa para ambas as classes. Mas isso gera um problema, veja a seguir.

Imagine que o nosso contexto atual é a de detecção de fraude de cartões de crédito. Para operar em um cenário de classificação regular, seriam necessários vários exemplos sobre as duas classes: “não-fraude” e “fraude”, sendo o nosso problema binário. O problema é que é difícil prever o que se encaixa na categoria de “fraude”, é difícil até mesmo ter dados que sejam comprovadamente fraude para treinar com os dados de “não-fraude”.

Algumas vezes é possível utilizar algumas técnicas para replicar, aumentar e equiparar o tamanho dos dados, neste contexto, de “fraude”. Esta técnica é conhecida como *oversampling*, e assim passa a utilizar algum algoritmo de classificação binária tradicional. Contudo, não é sempre possível garantir ou prever todos os padrões possíveis nesta abordagem.

Para os casos que apresentem esta disparidade tão grande de elementos dos dois conjuntos, podemos abrir mão do SVM de uma classe. O SVM de uma classe é representado pelo módulo chamado “One-Class Support Vector Machine”.

Neste cenário o modelo de SVM é treinado apenas com uma classe, sendo essa a classe regular. Seguindo o nosso exemplo seria a classe de não-fraude, justamente por ser muito difícil ter muitos exemplos catalogados de fraudes (ou invasão em redes, outras fraudes, ou outro comportamento anômalo).

Esta implementação, internamente, é um *wrapper* da biblioteca chamada *libsvm*<sup>45</sup>, uma das mais conceituadas e poderosas disponíveis.

Para configurar um SVM de uma classe são necessários quatro passos:

- Definição do modo de treino, encontra na opção *Create trainer mode: Single Parameter*; ou *Parameter Range*.
- *Single Parameter*: É para quando você sabe como você quer configurar o modelo. Neste caso você fornece um conjunto

---

<sup>45</sup> <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

de valores específicos para os parâmetros. Você pode ter essas informações de experimentos já realizados ou então fornecidos como referência em alguma fonte como artigos científicos, livros, periódicos, etc.

- *Parameter Range*: É para quando você não possui certeza sobre quais seriam os melhores parâmetros. Neste cenário você pode utilizar vários valores para os parâmetros e usar o módulo *Tune Model Hyperparameters* para escolher os melhores valores, construir/treinar e testar o seu modelo. O módulo *Tune Model Hyperparameters* itera sobre todas as possibilidades de combinações entre os valores para escolher o melhor conjunto.

Configurar os demais parâmetros, *nu* e *épsilon*.

- Parâmetro  $\eta$  (letra grega nu): Especifica o número da proporção, *ratio*, dos vetores de suporte. Determina o *trade-off* entre os casos considerados anomalias e os casos normais/regulares.
- Parâmetro  $\epsilon$  (letra grega épsilon): é utilizado como critério de parada da tolerância. Afeta o número de iterações usadas durante a otimização do modelo e dependerá do valor do critério de parada.
- Maiores detalhes sobre os parâmetros do SVM de uma classe podem ser encontrados na documentação<sup>46</sup>.
- Utilizar o módulo correto para treino. Isso dependerá da opção selecionada em Create Trainer mode.

Se a opção selecionada foi Single Parameter então utiliza o módulo Train Model para treino, da mesma maneira que seria em geral.

---

46 <https://msdn.microsoft.com/en-us/library/azure/dn913103.aspx>

Se a opção selecionada foi Parameter Range então você deve utilizar o módulo Tune Model Hyperparameters, anteriormente conhecido como Sweep Parameters. Maiores informações sobre este módulo podem ser encontradas na documentação<sup>47</sup>. Na sequência deve-se utilizar o módulo Train Anomaly Detection Model que efetivamente treinará o modelo utilizando os melhores parâmetros escolhidos.

- Em geral, após o módulo de *score model*, os scores não estarão calibrados/normalizados, sendo necessária também a utilização do módulo *Normalize Data* anexado a ele agindo sobre a coluna de *Scored Probabilities*. Pode-se também utilizar transformação Logística – *transformation method: Logistic*.

Sempre para os experimentos que são realizados, os parâmetros utilizados são fundamentais para termos um bom desempenho. Os parâmetros padrões, em geral, são uma boa primeira escolha. Eles não estão lá por acaso. São frutos de muitos anos de pesquisas e testes realizados com experimentos pelo time de pesquisa da Microsoft, conhecidos como *Microsoft Research*. No entanto, para facilitar o teste com vários elementos, utiliza-se o módulo *Tune Model Hyperparameters* que testa todas as combinações das variações dos parâmetros fornecidos.

Observe um trecho de um experimento utilizando *One-Class Support Vector Machine* com o *Tune Model Hyperparameters* na Figura 147.

---

47 <https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx>

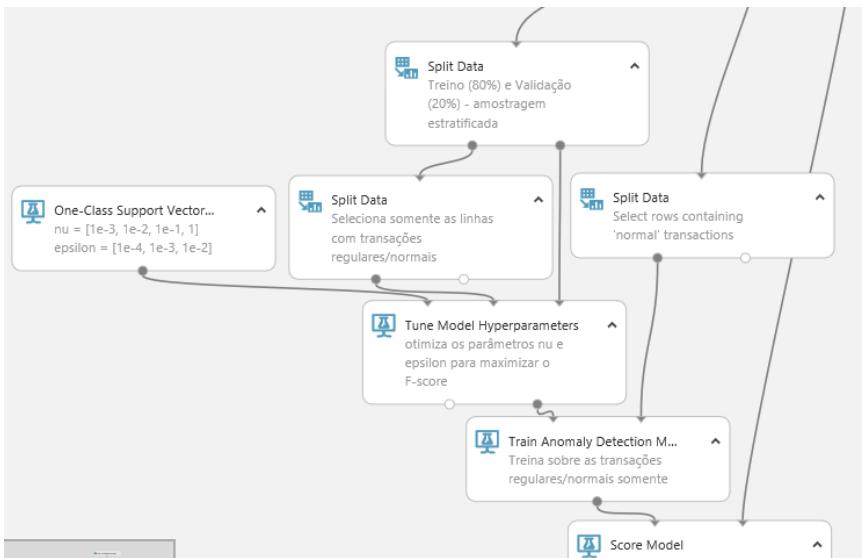


Figura 147 - Experimento utilizando One Class Support Vector Machine

O módulo *One-Class Support Vector Machine* cria um modelo baseado em um *kernel-SVM*, o que não é tão escalável, além de consumir bastante recursos físicos da máquina. Se o tempo de treino longo é uma restrição, ou a quantidade de dados é muito grande, recomendamos utilizar um outro método que faça a detecção de anomalias, como por exemplo o baseado em *PCA*.

## Detecção de anomalia baseado em Principal Component Analysis

O método baseado em *PCA* para detecção de anomalias é empregado nos mesmos cenários que o método baseado em *SVM* citado na seção anterior. Ou seja, quando é fácil a obtenção de dados para uma classe específica, mas difícil a obtenção de dados da classe anômala.

*PCA* é o acrônimo de *Principal Component Analysis*<sup>48</sup>, e é uma técnica de aprendizagem de máquina com bastante utilização na área de análise

---

48 A randomized algorithm for principal component analysis. <http://arxiv.org/abs/0809.2274>. Rokhlin, Szlam and Tygert.

exploratória de dados, pois mostra a estrutura dos dados bem como a sua variância. Esta técnica pode ser aplicada tanto para seleção de características, problemas de alta dimensionalidade<sup>49</sup> – maldição da dimensionalidade; quanto classificação, incluindo aqui a detecção de anomalias.

Esta técnica, para detecção de anomalias, separa um conjunto de casos, chamados de elementos de entrada, que contenham variáveis possivelmente correlacionadas no que chamamos de componentes principais, do inglês, *principal components*. O detector, para cada nova entrada, primeiro calcula a projeção dos autovetores, do inglês, *eigenvectors*, e depois calcula o erro de reconstrução normalizado.

Este erro de reconstrução normalizado é a distância quadrada entre o dado original e a sua estimativa, e que após calculado é o que consideramos como sendo valor de anomalia. Quanto maior o erro, mais anômalo é o elemento de entrada.

Os passos para utilizados para a utilização do módulo de *PCA-Based Anomaly Detection* são os mesmos utilizados pelo módulo One-class Support Vector, a diferença é que o módulo *Score Model* produzirá valores normalizados para os scores, o que remove a necessidade da utilização do módulo *Normalize Data*.

Desta forma os passos necessários para a sua utilização são:

- Especificar a forma de treino, através do Training mode.
  - Single Parameter, onde os parâmetros são fornecidos um conjunto específico de valores para os parâmetros. Utilize se já é de seu conhecimento os melhores parâmetros para o experimento em questão.

---

<sup>49</sup> Maldição da dimensionalidade (*curse of dimensionality*).  
[https://en.wikipedia.org/wiki/Curse\\_of\\_dimensionality](https://en.wikipedia.org/wiki/Curse_of_dimensionality)

- Parameter Range, neste caso são fornecidos conjuntos de valores para cada parâmetro. Utilize-o em conjunto ao Tune Model Hyperparameters.
- Defina os parâmetros para o PCA.
- *Number of components to use in PCA*: Especifica o total de componentes para se utilizar. Como guia geral não inclua um valor igual ao número de variáveis que você possua. O ideal é começar com um conjunto menor e aumentar conforme os critérios desejados. Utilize *Range Parameters* caso não esteja seguro sobre os melhores valores.
- *Oversampling parameter for randomized PCA*: Nos problemas de detecção de anomalias, o volume de elemento de classes diferentes é desbalanceado, o que dificulta a aplicação da técnica padrão de PCA. Ao especificar o *oversampling* é possível duplicar a instância alvo e executar o PCA sobre estes dados.
- *Enable input feature mean normalization*: Determina se os parâmetros de entrada serão normalizados ou não. Em geral é recomendado que haja a normalização, mantendo o valor padrão. Isto é devido ao fato do PCA realizar a maximização da variância entre as variáveis. Desmarque esta opção se os dados já estiverem normalizados.
- *Range for number of PCA components* e *Range for the oversampling parameter used in randomized PCA*: Utilize para especificar quantos valores serão incluídos em cada um dos parâmetros.

Maiores detalhes sobre os parâmetros do PCA para detecção de anomalias podem ser encontrados na documentação<sup>50</sup>.

---

50 <https://msdn.microsoft.com/en-us/library/azure/dn913102.aspx>

- Utilizar o módulo correto para treino. Isso dependerá da opção selecionada em *Create Trainer mode*:

Se a opção selecionada foi *Single Parameter* então utiliza o módulo *Train Model* para treino.

Se a opção selecionada foi *Parameter Range* então você deve utilizar o módulo *Tune Model Hyperparameters*. Maiores informações sobre este módulo estão na documentação<sup>51</sup>. Na sequência deve-se utilizar o módulo *Train Anomaly Detection Model* que efetivamente treinará o modelo utilizando os melhores parâmetros escolhidos.

Observe um trecho de um experimento utilizando *PCA-Based Anomaly Detection* com o *Tune Model Hyperparameters* na Figura 148.

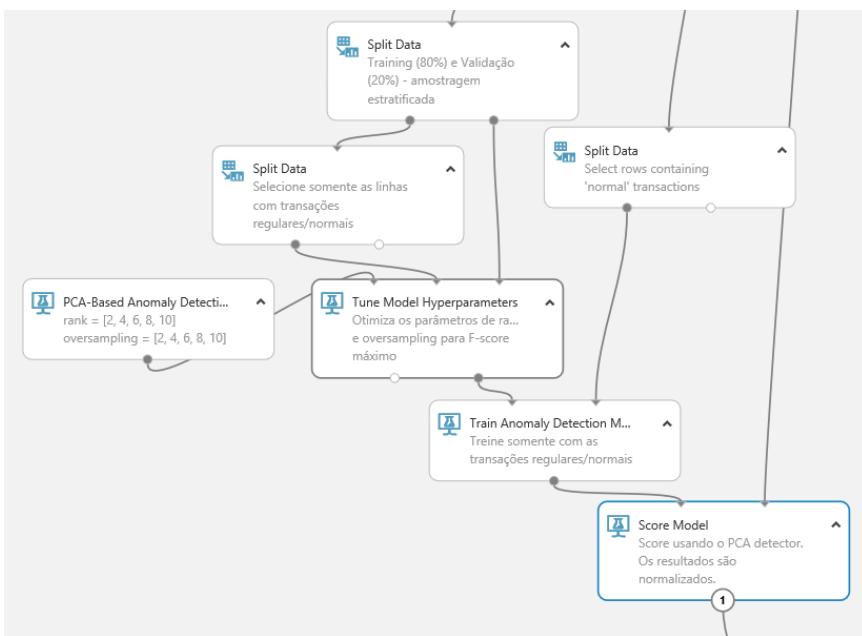


Figura 148 - Experimento utilizando PCA-Based Anomaly Detection Model

51 <https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx>

## Caminho das pedras na escolha do algoritmo

Para facilitar a escolha do algoritmo mais apropriado para a resolução do seu problema, veja algumas dicas nesta seção. É importante ressaltar que esta é uma ajuda para encontrar o algoritmo, não significa que ele deve continuar sendo utilizado quando você desconfiar que a predição pode ser melhorada por algum outro algoritmo, e ao testar, ele realmente melhorar o resultado.

Já foi discutido na primeira seção deste capítulo as diferenças dos grupos existentes no *Azure Machine Learning*. Com base nisso, você já deve ter uma ideia de qual tipo de algoritmo responderá suas perguntas.

Dentro do grupo de regressão, use esta tabela para auxiliar na sua tomada de decisão:

Descrição dos seus dados	Algoritmo proposto
Dados ranqueados em categorias ordenadas	Ordinal Regression
Contagem de eventos previstos	Poisson Regression
Prever a distribuição	Forest Quantile Regression
Treino mais rápido com um modelo linear	Linear Regression
Modelo linear para datasets pequenos	Bayesian Linear Regression
Boa acurácia, podendo demorar o treino	Neural Network Regression
Boa acurácia, treino mais rápido	Decision Forest Regression
Boa acurácia, treino mais rápido e grande rastro de memória	Boosted Decision Tree Regression

Dentro do grupo de Detecção de Anomalias, use esta tabela para escolha do algoritmo:

Descrição dos seus dados	Algoritmo proposto
Mais de 100 variáveis e fronteiras de decisão agressivas	One-Class SVM
Treino rápido	PCA - Based Anomaly Detection

Dentro do grupo de Classificação, ao se buscar multi classes, tenha esta tabela como base:

Descrição dos seus dados	Algoritmo proposto
Treino mais rápido com um modelo linear	Multiclass Logistic Regression
Boa acurácia, podendo demorar o treino	Multiclass Neural Network
Boa acurácia, treino mais rápido	Multiclass Decision Forest
Boa acurácia, pequeno rastro de memória	Multiclass Decision Jungle

Ainda no grupo de Classificação, porém focando em classificação binária:

Descrição dos seus dados	Algoritmo proposto
Mais de 100 variáveis e modelo linear	Two-class SVM
Treino mais rápido e modelo linear	Two-class Averaged Perceptron
Treino mais rápido e modelo linear	Two-class Logistic Regression
Treino mais rápido e modelo linear	Two-class Bayes Point Machine
Boa acurácia e treino mais rápido	Two-class Decision Forest
Boa acurácia, treino mais rápido e grande rastro de memória	Two-class Boosted Decision Tree
Boa acurácia, pequeno rastro de memória	Two-class Decision Jungle
Mais de 100 variáveis	Two-class Locally Deep SVM
Boa acurácia, podendo demorar o treino	Two-class Neural Network

As informações das tabelas encontradas nesta seção, foram retiradas do pôster do *Azure Machine Learning*<sup>52</sup>.

---

52 Pôster do Azure Machine Learning: <http://aka.ms/MLCheatSheet>

## 8 – Interagindo com Modelos Através de Notebooks

O trabalho do cientista de dados envolve muita experimentação bem como a necessidade de apresentar aos demais o raciocínio lógico empregado para alcançar os resultados obtidos. Dentre as alternativas existentes para tal, existem os *Jupyter Notebooks*.

Os *notebooks* são um conjunto de texto (explicativo, com definições ou o que mais seja julgado necessário), equações, parágrafos, links, códigos e resultados dos códigos (tabelas, resultados, gráficos). O AzureML suporta duas linguagens de programação para os códigos nos seus notebooks: R e Python (versão 2 e 3). O objetivo é ter um documento que seja legível e entendido por um humano, mas que contenha componentes que sejam entendidos também por uma máquina (por exemplo: modelos de aprendizagem de máquina).

No momento da escrita deste livro o recurso ainda se encontra em *preview*. Isto significa que alguns recursos podem ter algumas modificações na versão final, bem como adições podem ser realizadas até a sua publicação em GA (*general availability* – disponibilidade geral).

### Criando um notebook no AzureML

Através da opção no menu inferior “+ New” temos acesso à criação dos notebooks. Repare na Figura 149 que existem três opções básicas (*notebooks* em branco): Python 3, Python 2 e R.

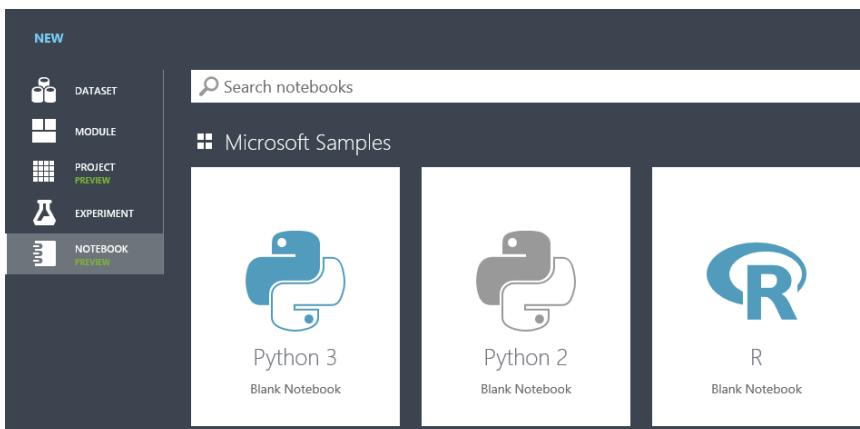


Figura 149 – Criar um novo notebook

Você irá perceber que existem modelos prontos para auxiliar em algumas tarefas comuns, entre elas: acessar dados de um experimento no AzureML, seleção de variáveis, avaliação de modelos, entre vários outros. Também é possível utilizar modelos disponíveis na galeria do AzureML, ou fazer o upload de um notebook próprio<sup>53</sup>).

Ao clicar para criar um *notebook* (iremos utilizar *notebooks* em R) é solicitado o nome do mesmo, conforme visto na Figura 150.

---

53 O notebook usado neste capítulo encontra-se disponível junto aos demais scripts disponibilizados digitalmente

## Name Notebook

NOTEBOOK NAME

Untitled 1



Figura 150 – Nomeie o notebook a ser criado

Após a sua criação é exibido, em uma nova página, um notebook em branco que aguarda pelos inputs que, como vimos, podem ser um texto explicando um raciocínio, um código (em R para o nosso caso), etc. A Figura 151 ilustra o notebook no seu estado inicial.

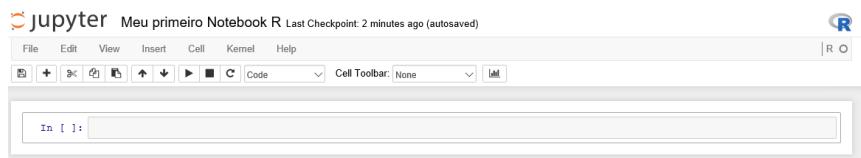


Figura 151 – Notebook R em branco

Um *notebook* é composto de células (*cells*). Cada célula possui um tipo: código, texto, cabeçalho, etc. Uma célula nova pode ser adicionada pelo botão adicionar ("+"), e o tipo da célula ativa pode ser alterado na lista suspensa, mostrada no exemplo como "Code" – código, ambos apresentados na Figura 151.

Nosso primeiro exemplo de notebook será para fazer a construção de um modelo de regressão linear simples, utilizando a linguagem R.

O conjunto de dados que iremos utilizar é o *mtcars*. Este *data frame* contém dados sobre carros, que foram extraídos da revista de 1974 chamada *Motor Trend* nos Estados Unidos. Mas, por que explicar aqui se podemos explicar no próprio notebook? Veja a Figura 152 retratar a base de dados a ser utilizada.

The screenshot shows a Jupyter Notebook interface with the title "Meu primeiro Notebook R" and a subtitle "Last Checkpoint: 31 minutes ago (autosaved)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Help, and various cell type icons. A dropdown menu shows "Cell Toolbar: None". The main area has a section titled "Análise de Carros" with the following text:  
Para adicionar o texto em destaque acima, utilizamos a seguinte entrada (sem as aspas) "## Análise de Carros" e depois Shift + Enter.  
Para este exemplo utilizaremos um dataframe padrão do R chamado mtcars. Veja abaixo maiores informações!  
In [1]: ?mtcars  
mtcars (datasets) R Documentation  
The code cell contains "?mtcars" and shows the help documentation for the mtcars dataset. Below the code cell, there is a section titled "Motor Trend Car Road Tests" with "Description", "Usage", and "mtcars".

Figura 152 – Primeiros passos do nosso notebook

Para a construção do pequeno notebook mostrado na Figura 152, utilizamos duas células: a primeira do tipo *markdown* para os textos e a segunda do tipo *code*. Para visualizar o resultado do código inserido deve-se clicar no botão *play* que executa o código da célula. Também é possível executar uma célula com as combinações: <Shift> + <Enter>.

É uma excelente maneira de demonstrar o raciocínio empregado passo a passo. Repare que o código neste exemplo é apenas *?mtcars* que retorna a página de ajuda deste *data frame*.

Para adicionar uma marcação de cabeçalho (alterar o tamanho da fonte) utilize o caractere "#" no início da linha desejada, e, depois de entrar com o texto, execute o processamento da célula. Com isso o resultado deve ser semelhante ao exibido na Figura 152.

A célula *markdown* pode ter seu conteúdo formatado, utilizar negrito, itálico e texto tachado/riscado, bem como listas, tabelas, links, imagens, e até formulas em *LaTEX*<sup>54</sup>.

Para adicionar um link para um site externo, a sintaxe fica como mostrado a seguir.

```
[link](www.link.com)
```

Ou você pode utilizar a própria sintaxe HTML, algo como:

```
<a href="www.link.com">texto do link</a>
```

Para adicionar expressões *inline* utilize a expressão entre \$. Para adicionar expressões em uma linha própria utilize a expressão entre \$\$ . Na Figura 153 é possível acompanhar um exemplo com formatação, expressões matemáticas e links. E na Figura 154 encontra-se o resultado obtido após executar a célula em questão.

```
# Cabeçalho - Formatação
## Cabeçalho
### Cabeçalho

**Negrito**, *itálico*, ~~riscado~~

Expressão mate mática no meio de outro texto: ${3 / 1} * e^{(i)} + 5 * \pi = 0$ 

Em uma única linha e aumentando com um "#":
# $e^x=\sum_{i=0}^{\infty} \frac{1}{i!}x^i$
```

Exemplo de função retirado de [\[link\]](http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Working%20With%20Markdown%20Cells.html) (<http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Working%20With%20Markdown%20Cells.html>)

Figura 153 – Código para formatar o texto na célula

---

54 Projeto LaTEX: <http://www.latex-project.org/about/>

## Cabeçalho - Formatação

### Cabeçalho

#### Cabeçalho

**Negrito**, *ítalo*, riscado

Expressão mate mática no meio de outro texto:  $3/1 * e^i + 5 * \pi = 0$

Em uma única linha e aumentando com um "#".

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

Exemplo de função retirado de [link](#)

Figura 154 – Resultado da formatação de texto no notebook

Dicas e exemplos para formatação da célula *markdown* você pode encontrar na documentação<sup>55</sup> e na web<sup>56</sup>.

Neste momento temos o conhecimento para explicar o procedimento utilizado (de maneira bem formatada), e também de como executar códigos em R (para Python o processo é o mesmo).

O data frame *mtcars* contém duas colunas de nosso interesse: *wt* (peso) e *mpg* (milhas por galão – consumo de combustível). Vamos criar um modelo simples (regressão linear) que tenta explicar o consumo de combustível baseado no peso do carro.

Em R a expressão necessária é a seguinte:

---

55

<http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Working%20With%20Markdown%20Cells.html>

56

<https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet>

```
lm(mtcars$mpg ~ mtcars$wt)
```

Veja na Figura 155 esse código sendo executado.

```
In [2]: mpg <- mtcars$mpg  
wt <- mtcars$wt  
  
m <- lm(mpg ~ wt)  
m
```

Call:  
lm(formula = mpg ~ wt)

Coefficients:  
(Intercept)                   wt  
             37.285              -5.344

Figura 155 – Código para calcular a regressão linear (“wt” explicando “mpg”)

É possível visualizar gráficos do R como saída no *Jupyter*. Isso facilita muito a compreensão por parte de quem está vendo a saída do notebook. A Figura 8 ilustra o resultado da regressão linear mostrada na Figura 155.

A função *plot* utilizada é representada a seguir. Perceba que utilizamos algumas formatações para nome do gráfico e dos eixos, bem como cor, etc.

```
plot(mtcars$wt, mtcars$mpg, pch = 16, cex = 1.3, col = "red", main = "Consumo (mpg x  
wt)", xlab = "Peso (wt)", ylab = "Consumo (mpg)")  
  
abline(m[[1]][1], m[[1]][2])
```

A função *abline* plota uma linha sobre o gráfico atual. Ela necessita do valor de intersecção com o eixo Y bem como o coeficiente angular da reta. Ambos fornecidos pela função *lm*, que no nosso caso são os valores: 37.285 e -5.344 respectivamente. O resultado desta função é uma lista onde o primeiro valor é um vetor com duas posições, cada uma com um coeficiente da reta. Para o exemplo ficar mais genérico utilizamos as referências ao invés dos valores diretamente. Repare também que o objeto *m* foi criado em outra célula, mas ainda assim é acessível nesta célula posterior.

```
In [7]: plot(mtcars$wt, mtcars$mpg, pch = 16, cex = 1.3, col = "red", main = "Consumo (mpg x wt)",  
         xlab = "Peso (wt)", ylab = "Consumo (mpg)")  
abline(m[[1]][1], m[[1]][2])
```

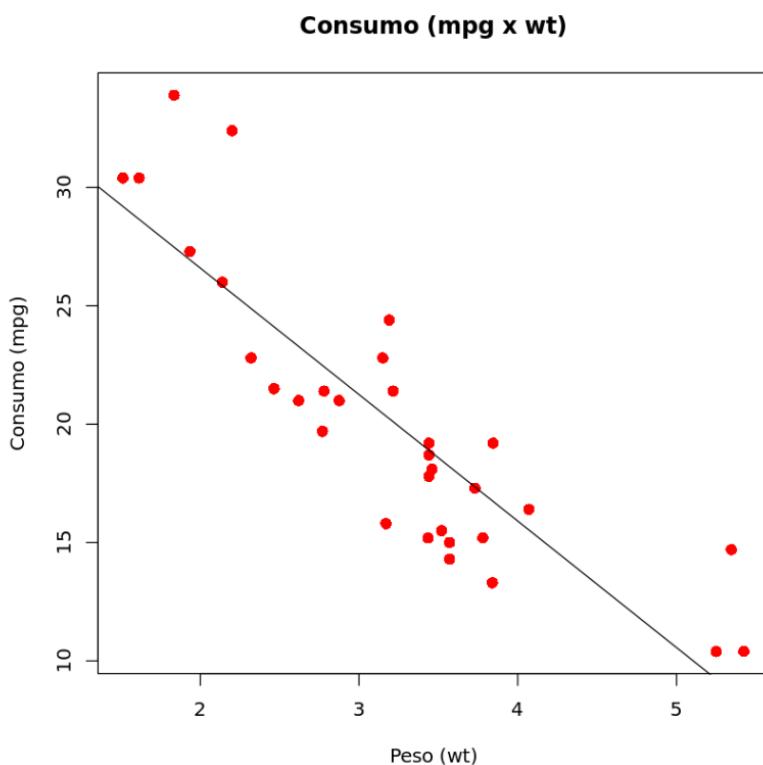


Figura 156 – Gráfico de consumo de combustível em função do peso (mtcars)

Como resultado da nossa pequena análise temos que os carros com menor peso, possuem uma autonomia melhor (maior quantidade de milhas rodadas por galão de combustível). Perceba que essa é uma análise bastante simples, mas ilustra o poder e a flexibilidade dos notebooks.

É possível exportar a análise realizada nos notebooks para diferentes formatos, seja para utilizar em outro local, arquivar, imprimir, ou outra finalidade. A Figura 157 mostra as opções disponíveis.

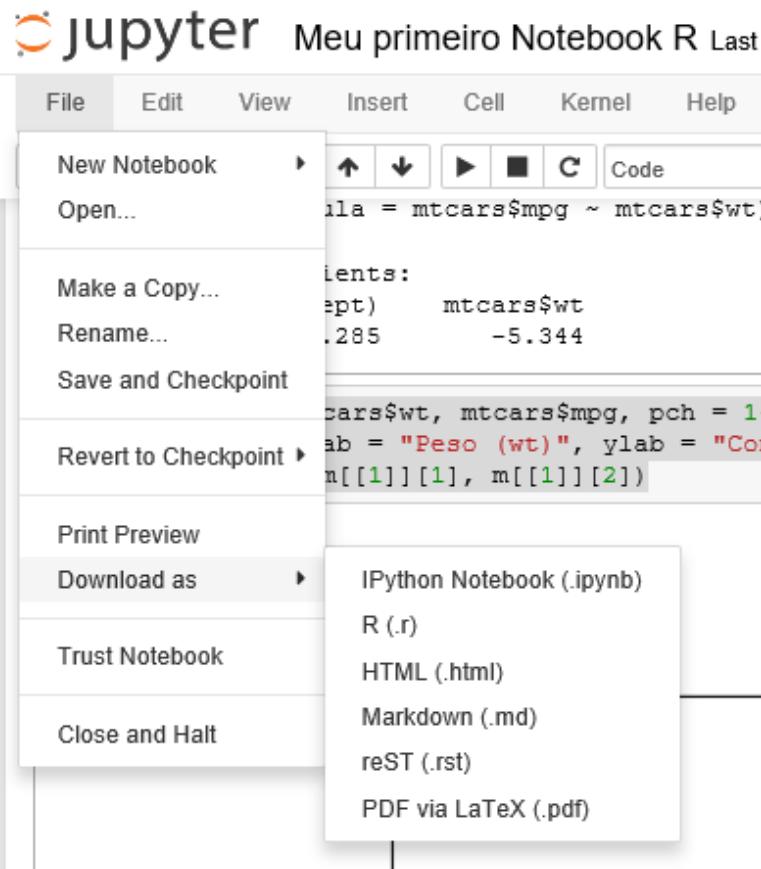


Figura 157 – Formatos de exportação de um notebook no AzureML

Existem diversos outros recursos que fogem ao escopo deste livro, mas há a possibilidade de utilizar outros recursos do próprio *Jupyter*, como os *checkpoints* (gravações graduais), a possibilidade de reverter a um estado anterior, etc.

Para identificar a versão atualmente executada pelo *Jupyter* basta ir em *Help >> About*. Acompanhe esta opção na Figura 158.

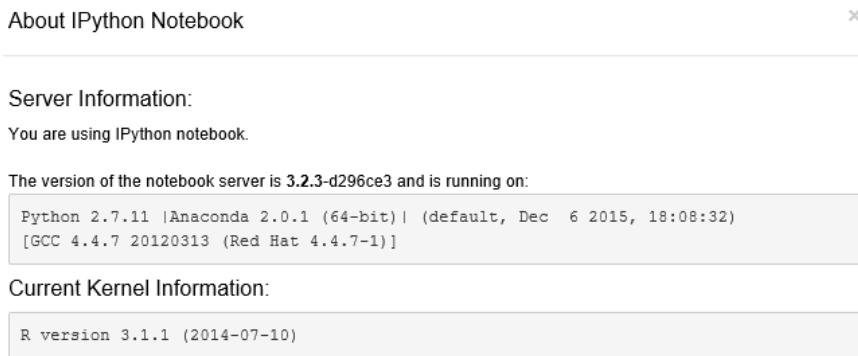


Figura 158 – Tela de “sobre” com os dados de versão do R e Python do notebook atual

## Integração dos Notebooks ao AzureML

O *Jupyter* pode ser utilizado inteiramente apartado de um experimento no *AzureML*, conforme vimos na seção anterior. Contudo, para a sua operacionalização e também para aproveitar os recursos e facilidades da plataforma, é possível integrar os códigos executados pelo notebook com os experimentos do *AzureML*.

Após a criação de um notebook é possível compartilhá-lo na galeria (*Cortana Intelligence Gallery*) de forma pública ou privada.

É possível fazer download ou upload de *datasets*, baixar *datasets* intermediários (de algum ponto específico do experimento) e também publicar ou consumir *web services*.

Por padrão, existe um pacote chamado “*AzureML*” já instalado nos servidores de *Jupyter* do *AzureML*. Este pacote contém as implementações necessárias para realizarmos as integrações entre o *notebook* e o serviço em nuvem.

A função necessária para configuração do seu notebook para esta integração é a *workspace*.

É necessário apresentar suas credenciais ao se conectar no *AzureML*. Contudo, isso poderia expor as suas credenciais e gerar um problema de segurança ao compartilhar um notebook. Pensando nisso a Microsoft disponibilizou dentro do seu *workspace* as credenciais em um arquivo *.json*.

Este arquivo fica localizado em *~/.azureml/settings.json* no storage que faz parte do *AzureML*<sup>57</sup> configurado no Portal do Azure, e como fica apenas a referência no código do seu notebook, não expõe estas informações no caso de compartilhamento.

Outro ponto importante é que ao executar o notebook dentro da plataforma, a função *workspace* não necessita de credenciais, herdando as que foram utilizadas no login, caso execute o notebook na sua máquina local ou outro servidor com o *Jupyter* instalado, serão necessárias as mesmas (*workspace\_id* e *authorization\_token*, obtidos no portal do *AzureML*). O código a seguir ilustra uma lógica de teste para fornecer as credenciais.

```
library("AzureML")
if(file.exists("~/azureml/settings.json")){
  ws <- workspace()
} else {
  # workspace_id <- ""
  # authorization_token <- ""
  ws <- workspace(workspace_id, authorization_token)
}
```

O *workspace\_id* e o *authorization\_token* podem ser coletados conforme mostra a Figura 159 e Figura 160, trate-os com sigilo, eles são as

---

57 Pacote *AzureML*: [https://cran.r-project.org/web/packages/AzureML/vignettes/getting\\_started.html](https://cran.r-project.org/web/packages/AzureML/vignettes/getting_started.html)

sus senhas. Para acesso, ambos os *tokens* de autorização (*authorization tokens*) são aceitos.

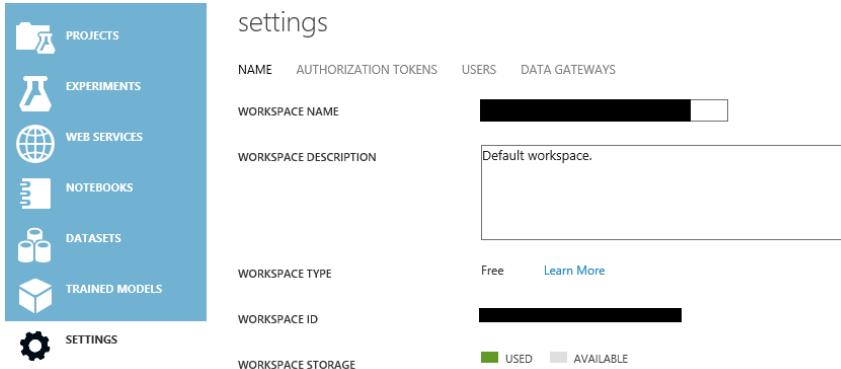


Figura 159 – Recuperar o Workspace ID

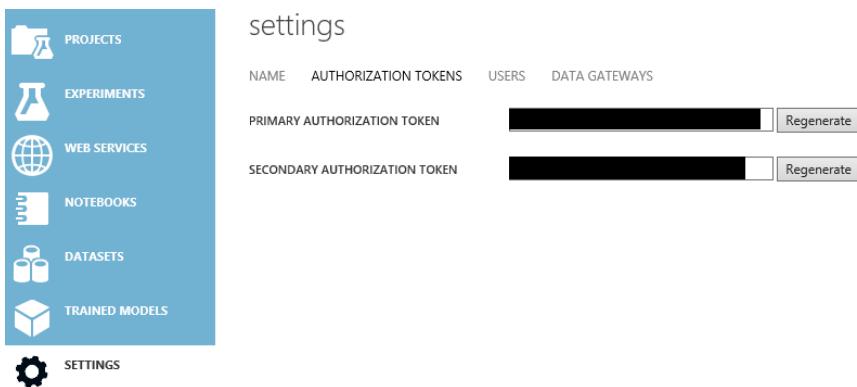


Figura 160 – Recuperar o Authorization Token

Também é permitida a instalação de novos pacotes, disponíveis no MRAN (*Microsoft CRAN Mirror*).

Pode-se utilizar a função *install.packages*, similar ao R tradicional. Uma sugestão é utilizar a seguinte construção para instalar o pacote desejado.

Repare que no exemplo a seguir estamos fazendo a instalação do pacote chamado corrgram<sup>58</sup>.

```
if(!require("corrgram", quietly = TRUE))  
install.packages("corrgram")
```

## Interagindo com um dataset do AzureML

Para utilizar um dataset do AzureML em seu notebook é necessária a utilização da função `download.datasets` ou `download.intermediate.dataset`.

Observe o trecho de um notebook, apresentado na Figura 161, que realiza o download do dataset chamado *Iris Two Class Data*.

A partir deste momento é possível aplicar as mesmas técnicas explicadas na seção anterior, ou seja, scripts em R (ou Python) diretamente no *notebook* e ainda assim aproveitando um conjunto de dados já definido no *AzureML*.

---

58 MRAN – Corrgram: <https://mran.microsoft.com/package/corrgram/>

```
In [1]: library("AzureML")

if(file.exists("~/azureml/settings.json")){
  ws <- workspace()
} else {
  # workspace_id <- ""
  # authorization_token <- ""
  ws <- workspace(workspace_id, authorization_token)
}

iris_df <- download.datasets(ws, "Iris Two Class Data")
head(iris_df)
```

	Class	sepal-length	sepal-width	petal-length	petal-width
1	1	6.3	2.9	5.6	1.8
2	0	4.8	3.4	1.6	0.2
3	1	7.2	3.2	6	1.8
4	0	5.2	3.4	1.4	0.2
5	1	6.7	3.1	5.6	2.4
6	0	4.9	3.6	1.4	0.1

Figura 161 – Lendo um conjunto de dados do AzureML no notebook

Outra possibilidade é criar um novo notebook com acesso a dados intermediários do seu experimento. Para tal basta adicionar um módulo chamado *Convert to CSV* e uma das opções presente no seu menu da saída será para abrir em um notebook (R ou Python). Observe o resultado nas Figuras 162 e 163.

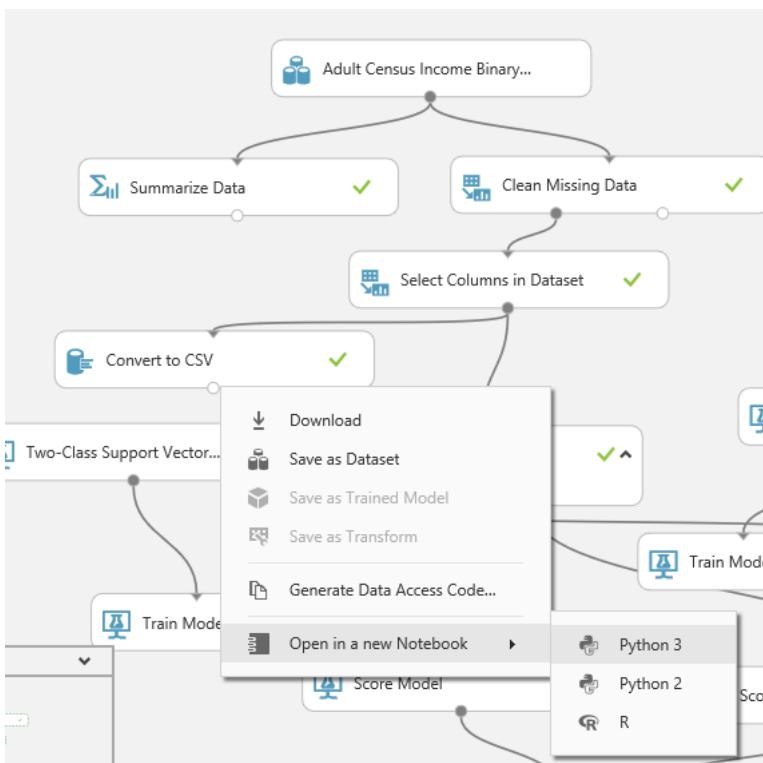


Figura 162 – Menu para criar um notebook com um conjunto de dados intermediários

```
In [1]: library("AzureML")
ws <- workspace()
dat <- download.intermediate.dataset(
  ws,
  experiment = "████████████████████████████████",
  model_id = "████████████████████████████████",
  port_name = "Results dataset",
  data_type_id = "GenericCSV"
)

In [2]: head(dat)
```

	age	education	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
1	39	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
2	50	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
3	38	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
4	53	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
5	28	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
6	37	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K

Figura 163 – Novo notebook criado com acesso aos dados intermediários do experimento

## Publicando Web Services

Como mencionado, é possível utilizar os notebooks para publicar e consumir *web services*, e assim operacionalizar o experimento de *machine learning*. Isso é possível mesmo que tenha a sua origem em um notebook ao invés de um experimento do *AzureML* propriamente dito.

Para este exemplo iremos utilizar o modelo linear que criamos em uma das seções anteriores, relembrar a regressão linear com a Figura 156, e criar um *web service* para expô-lo e poder operacionalizar o processo.

```
m <- lm(mpg ~ wt)
```

Vimos também que este modelo (armazenado na variável *m*) pode ser utilizado em células seguintes. Caso, você esteja reproduzindo os exemplos e esteja utilizando o mesmo notebook para todos, não será necessário recriar o modelo. Neste caso basta referenciar a variável *m*.

**Importante:** para funcionar o que é mencionado no parágrafo anterior, é necessário que tenha sido executado na sessão atual o script que

criou a variável ("m"). O AzureML não irá manter na memória o objeto uma vez que a sessão com o notebook seja encerrada.

A função que utilizamos para publicar um web service é a: *publishWebService*. Antes da publicação efetiva do Web Service, devemos criar uma função que testa nossa previsão. Esta função será necessária a sua publicação. Veja o código completo na Figura 164.

```
# função de predição
funcao_predict <- function(dados){
  predict(m, dados) # "m" é a variável com o modelo linear já criado
}
# um pequeno subconjunto de mtcars para testar a função
primeiros_cinco <- mtcars[1:5,]

# testar a função
data.frame( real = primeiros_cinco$mpg,
            previsto = mypredict(primeiros_cinco))
```

```
In [5]: library(AzureML)
if(file.exists("~/azureml/settings.json")){
  ws <- workspace()
} else {
  #ws <- workspace(workspace_id, authorization_token)
}

# função de predição
funcao_predict <- function(dados){
  predict(m, dados) # "m" é a variável com o modelo linear já criado
}

# um pequeno subconjunto de mtcars para testar a função
primeiros_cinco <- mtcars[1:5, ]

# testar a função
data.frame( real = primeiros_cinco$mpg,
            previsto = funcao_predict(primeiros_cinco))
```

	<b>real</b>	<b>previsto</b>
<b>Mazda RX4</b>	21	23.28261
<b>Mazda RX4 Wag</b>	21	21.91977
<b>Datsun 710</b>	22.8	24.88595
<b>Hornet 4 Drive</b>	21.4	20.10265
<b>Hornet Sportabout</b>	18.7	18.90014

Figura 164 – Predição dos valores de acordo com o modelo de regressão linear

Na sequência fazemos a publicação do web service (e criação do endpoint). Acompanhe este processo na Figura 165. A execução da função *str()*, que retorna a estrutura e dados, é opcional e pode ser omitida. Importante observar que ela retorna valores sensíveis, como o Workspace Id, PrimaryKey entre outros. Utilize apenas em fase de validação.

```
In [9]: # publicar o webservice (publishWebService())
endpoint <- publishWebService(ws = ws, # workspace
                                fun = funcao_predict,
                                name = "MPGxWT-PredicaoDeConsumo",
                                inputSchema = primeiros_cinco)

str(endpoint)

Classes 'Endpoint' and 'data.frame': 1 obs. of 13 variables:
 $ Name          : chr "default"
 $ Description   : chr ""
 $ CreationTime  : chr "2016-09-15T10:40:52+00:00"
```

Figura 165 – Publicação de um webservice no AzureML com um modelo criado no notebook

Com isso é possível observar que este webservice passa a ser disponível da mesma maneira que os outros webservices já criados (através da aba de webservices no AzureML), conforme observado na Figura 166.

NAME	CREATED ON	PROJECT
MPGxWT-PredicaoDeConsumo	8/15/2016 10:40:52 AM	None

Figura 166 – Web service criado no notebook exposto através do AzureML

Todas as vezes que o trecho de código contendo a função `publishWebService` executar, será criado um novo web service. Caso deseje atualizar informações sobre um web service existente, utilize a função: `updateWebService`. Para remover um web service *programaticamente* no notebook utilize a função `deleteWebService`.

## Consumindo um Web Service

Existem casos em que se necessitamos utilizar modelos já criados em outros momentos, dentro de um notebook em que você esteja trabalhando.

Um exemplo clássico é durante o trabalho sobre as características, encontrada na literatura como *feature engineering*, em que podemos utilizar outros modelos preditivos já criados para ajustar nossos padrões de entrada. Entre outras atividades podemos citar o preenchimento de valores faltantes, conhecidos como *missing values*, com valores mais complexos, ao invés de valores fixos, médias, medianas, etc.

Tendo isso em mente, o consumo de *web services* através de um notebook em que esteja trabalhando também se torna necessário. O *AzureML* possibilita consumi-los dentro de notebooks para este fim.

O webservice a ser consumido pode ser originário da mesma sessão, ou de outro *web service* já publicado no *AzureML*.

Para utilizar um *web service* criado na mesma sessão, basta fazer referência ao *endpoint* (no nosso caso, a variável se chama *endpoint*). Ainda que esta opção seja possível, é mais rápido e comumente utilizado a chamada diretamente na função de predição que já existe no notebook. Para utilizar um *web service* de outra sessão, devemos gravar algumas informações sobre o mesmo para poder utilizar em outra seção, inclusive em outros notebooks.

A função de consumo de um *web service* é a *consume*. E na sua utilização mais simples devemos passar como parâmetros o *endpoint* e o conjunto de dados de entrada. Repare que a chamada ao webservice pode necessitar de várias tentativas, conhecidas como *retry*. A sessão também deve, obrigatoriamente, estar conectada com credenciais de um usuário válido do *AzureML*. O resultado que nos interessa da chamada do *consume* é a coluna *ans*, podendo ser obtida através da chamada:

```
consume(endpoint, dataframe)$ans
```

Maiores detalhes podem ser obtidos na documentação ao executar o comando *?consume*.

Observe nas Figuras 167 e 168 os exemplos consumindo o nosso *web service* publicado na seção anterior, utilizando o *endpoint* da mesma seção e criando um oriundo de outra seção, respectivamente. A função *consume* deve ser usada para consumo em ambos os casos.

```
In [ ]: #testar com o mesmo conjunto utilizado anteriormente  
result <- consume(endpoint, primeiros_cinco)$ans  
data.frame(real = primeiros_cinco$mpg, previsto = result)
```

Figura 167 - Chamada para consumo de um web service

## Análise preditiva com Azure Machine Learning e R

```
In [8]: # É necessário estar logado em uma sessão válida do AzureML
# Existem duas alternativas para recuperar as credenciais de acesso
# Alternativa 1: Criar um arquivo temporário para armazenar os dados do endpoint
arquivo <- tempfile(fileext = ".rds")
saveRDS(endpoint, file = arquivo)

# Leia os dados Read the endpoint data from file
endpoint <- readRDS(arquivo)

# # Alternativa 2: Grave seu workspace ID, authorization token e webservice ID
# vs_id <- vs$nid
# vs_auth <- vs$auth
# service_id <- ep$WebServiceId

# # Defina o workspace, isso é necessário caso esteja executando fora da sessão que criou o serviço
# ws <- workspace(
#   id = vs_id,
#   auth = vs_auth
# )
# # Defina o endpoint baseado no workspace e service ID information
# endpoint <- endpoints(ws, service_id)
```

```
In [9]: str(endpoint)

Classes 'Endpoint' and 'data.frame': 1 obs. of 13 variables:
 $ Name           : chr "default"
 $ Description    : chr ""
```

Figura 168 - Acesso a um web service de outra sessão

## 9 – Aumentando as Possibilidades com R

A Linguagem R é uma linguagem estatística, open-source, e usada em larga escala em cenários de aprendizagem de máquina. O objetivo é ser uma linguagem para manipulação de dados, cálculos e exibição gráfica. Muito se deve ao fato da simplicidade do uso para a construção de modelos matemáticos, estatísticos e gráficos.

Foi criada por Ross Ihaka e Robert Gentleman do departamento de Estatística da universidade de Auckland, Nova Zelândia, e pelo fato de ser open-source recebeu e recebe diversas contribuições de desenvolvedores ao redor do mundo.

Outro aspecto que contribui para a grande popularidade do R é o fato de ser multi plataforma, ou seja, ele está disponível para Windows, (Mac) OS X e Linux.

R é sem dúvida uma das linguagens de programação mais utilizadas no mundo quando tratamos de aprendizagem de máquina. Recentemente tem atraído bastante atenção de grandes corporações, como a Microsoft que adquiriu a empresa Revolution Analytics e lançou uma nova linha de produtos denominada de Microsoft R Server, junto a isso a incorporação das suas capacidades de processamento paralelo e escalabilidade (originárias da Revolution Analytics) dentro do SQL Server 2016 (SQL Server R Services). Além do fato da inclusão do R em diversos outros produtos como o Power BI<sup>59</sup> e o próprio AzureML tratado neste livro.

### IDE para desenvolvimento

---

59 URL do Power BI: [www.powerbi.com](http://www.powerbi.com)

Existem diversas IDEs para desenvolvimento dos scripts em R. As duas mais populares são: o R Studio<sup>60</sup>, com certeza é uma das mais utilizadas, onde a versão desktop *open source* é gratuita, e o R Tools for Visual Studio<sup>61</sup>, também gratuito.

O objetivo deste módulo não é cobrir todos os aspectos da linguagem R, na realidade, para isso, seria necessário um livro inteiramente dedicado a esta linguagem. Os subcapítulos a seguir cobrem alguns aspectos e conceitos importantes para o entendimento dos exemplos em R expostos ao longo deste livro, bem como o que é necessário para uma compreensão básica da linguagem.

Antes de começarmos é importante conhecermos a fonte onde é possível fazer o download do R e de seus pacotes.

O CRAN (*Comprehensive R Archive Network*) são servidores que armazenam o R e sua documentação ao redor do mundo. É onde podemos fazer o download do R para utilização junto a sua IDE de escolha. O CRAN<sup>62</sup> também funciona como repositório dos pacotes desenvolvidos para a plataforma..

Existem outros repositórios similares, como o MRAN<sup>63</sup> da Microsoft, que podem oferecer snapshots de versões diferentes dos pacotes publicados no CRAN.

## Utilizando a Linguagem R – Conceitos e Estruturas

Os scripts em R são interpretados e executados linha a linha. Não há uma extensão oficial, mas em geral usamos .R para scripts e .RData para dataframes armazenados em arquivos. Não é necessário criar um arquivo para executar os seus códigos, você pode escrevê-los, analisar os resultados e

---

60 URL do RStudio: [www.rstudio.com](http://www.rstudio.com)

61 URL do R Tools for Visual Studio: <https://www.visualstudio.com/vs/rtvs/>

62 URL do CRAN: <https://cran.r-project.org/>

63 URL do MRAN: <https://mran.microsoft.com>

então executar um comando novo de acordo com a sua necessidade. Na realidade, no processo exploratório de dados realizado por um cientista de dados esta abordagem é a mais comumente utilizada.

## Expressões Aritméticas e Strings

O R consegue lidar com expressões aritméticas e strings.

Ao executarmos por exemplo:  $10 + 10$  ou  $3 * 5$  obtemos a respostas conforme a seguir.

*Você pode testar estes códigos (ou similares) no seu próprio ambiente, basta instalar o R (download no CRAN) e, preferencialmente, uma IDE para facilitar.*

```
> 10 + 10
```

```
[1] 20
```

```
> 3 * 5
```

```
[1] 15
```

Também é possível lidar com strings, para tal necessitamos utilizar aspas duplas para referenciá-las.

```
> "Diego e Thiago"
```

```
[1] "Diego e Thiago"
```

Expressões lógicas retornando valores booleanos também são suportados.

```
> 3 + 17 == 20
```

```
[1] TRUE
```

Um atalho/apelido para TRUE (verdadeiro) e FALSE (falso) são as letras T e F.

```
> FALSE == F
```

```
[1] TRUE
```

```
> T == TRUE
```

```
[1] TRUE
```

## Variáveis

Da mesma forma que em outras linguagens de programação, temos a possibilidade de armazenar os valores resultantes de expressões e funções em variáveis para uso posterior. No R o operador de atribuição é o "<->" (sem as aspas) ou o sinal de igualdade "=" (também sem as aspas). No exemplo a seguir armazenamos o resultado de uma expressão na variável "v" e depois podemos utilizar "v" em outras operações subsequentes, incluindo atribuir a outras variáveis (no exemplo, a variável "x").

O conteúdo de uma variável pode ser substituído por outros valores a qualquer tempo, mesmo que os valores sejam de um tipo de dados diferente.

*O valor resultante não será mostrado após a atribuição a uma variável. Por isso necessitamos "executar" o nome da variável para vermos seu conteúdo.*

```
> v <- 10 + 342  
> v  
[1] 352  
  
> x <- v / 2  
  
> x  
[1] 176  
  
> x <- "Diego e Thiago"  
  
> x  
[1] "Diego e Thiago"
```

## Funções

As funções podem ser originadas do R ou de pacotes instalados por você originados da comunidade (veremos como fazer isso posteriormente).

Para executar uma função utilizamos o seu nome seguido de seus parâmetros, se necessários.

A função sum, executa a soma de diferentes valores:

```
> sum (43, 1, -11)  
[1] 33  
  
> x <- 3  
  
> sum (43, 1, x)
```

[1] 47

Existem algumas funções que ajudam inclusive a ver outros exemplos do R. Um exemplo é a função `demo()`. A função `help` provê acesso a documentação sobre um componente ou função. Experimente o seguinte comando no RStudio.

```
> help(demo)
```

Será aberto a referência para a função `demo`, conforme pode ser observado na Figura 169.

The screenshot shows the RStudio interface with the 'Help' tab selected. The main content area displays the 'R: Demonstrations of R Functionality' page. At the top left, there's a search bar and a 'Find in Topic' button. Below the search bar, the text 'demo {utils}' is shown. The page title is 'Demonstrations of R Functionality'. Under the title, there's a 'Description' section which states: 'demo is a user-friendly interface to running some demonstration R scripts. `demo()` gives the list of available topics.' Below that is a 'Usage' section with the following code snippet:

```
demo(topic, package = NULL, lib.loc = NULL,  
      character.only = FALSE, verbose = getOption("verbose"),  
      echo = TRUE, ask = getOption("demo.ask"),  
      encoding = getOption("encoding"))
```

Under the 'Usage' section is an 'Arguments' table:

topic	the topic which should be demonstrated, given as a <a href="#">name</a> or literal character string, or a character string, depending on whether <code>character.only</code> is <code>FALSE</code> (default) or <code>TRUE</code> . If omitted, the list of available topics is displayed.
package	a character vector giving the packages to look into for demos, or <code>NULL</code> . By

Figura 169 - Exemplo do resultado da função `help()`

Existem atributos que possuem nomes, neste caso para especificá-los é necessário utilizar o seu nome seguido de “=” (sem as aspas), conforme no exemplo a seguir.

```
> rep("Thiago", times = 4)  
[1] "Thiago" "Thiago" "Thiago" "Thiago"
```

Mais informações sobre a função *rep*? Tente *help(rep)*.

Gostaria de alguns exemplos do uso de uma função para conhecê-la melhor? Tente *example(<nome da função>)*. A seguir temos o início do retorno desta função.

```
> example(rep)  
  
rep> rep(1:4, 2)  
[1] 1 2 3 4 1 2 3 4  
  
rep> rep(1:4, each = 2)    # not the same.  
[1] 1 1 2 2 3 3 4 4  
  
rep> rep(1:4, c(2,2,2,2))  # same as second.  
[1] 1 1 2 2 3 3 4 4
```

```
rep> rep(1:4, c(2,1,2,1))  
[1] 1 1 2 3 3 4  
  
rep> rep(1:4, each = 2, len = 4) # first 4 only.  
[1] 1 1 2 2  
...
```

## Executando Arquivos

Para scripts curtos ou análise exploratória, as entradas diretas de comandos como os listados anteriormente são bastante indicadas. Caso o desejado seja facilitar a reprodução de código, bem como a execução de sequências já conhecidas o indicado é a utilização de arquivos com os códigos. Por convenção usamos a extensão .R, mas não é obrigatória.

O R trabalha com um conceito de Diretório de Trabalho (working directory). Quando não especificamos um local, assumimos que estaremos fazendo referência ao local do diretório de trabalho atual. Para alterar este local utilizamos a função setwd (o diretório precisa existir).

```
> setwd("D:/temp")
```

Para listar o conteúdo deste diretório, para localizar e ver o nome dos scripts em R podemos utilizar a função list.files(). Para executar um script localizado neste diretório utiliza a função source() conforme o exemplo. O parâmetro print.eval é usado para retornar os resultados impressos no script. Caso o desejado seja mostrar o fonte antes da execução de cada linha do script, podemos usar a opção echo = TRUE.

```
> source("ExibeTexto.R", print.eval = T)
[1] "Texto mostrado pela execução do arquivo chamado ExibeTexto.R"
> source("ExibeTexto.R", echo = T)

> x <- "Texto mostrado pela execução do arquivo chamado ExibeTexto.R"

> x
[1] "Texto mostrado pela execução do arquivo chamado ExibeTexto.R"
```

## Vetores

A estrutura de dados mais simples do R é um vetor. Um vetor pode ser criado com a função “c”. C é o diminuto para a função combine, que consegue combinar listas de valores e gerar um vetor.

Os vetores podem ser de valores numéricos, lógicos, texto, etc., desde que todos os membros de um mesmo vetor sejam do mesmo tipo de dados. Caso você tente criar um vetor com tipos de dados diferentes, eles serão convertidos para uma mesma base (string/texto, por exemplo) e assim o vetor é criado (atenção: nenhuma mensagem de erro ou alerta é mostrada).

Por exemplo, para criar um vetor com os valores de 1 a 5 podemos fazer:

```
> c(1,2,3,4,5)
[1] 1 2 3 4 5
> c(1:5)
```

```
[1] 1 2 3 4 5
```

Para facilitar a criação de intervalos contínuos, sem a necessidade de especificar todos os valores, podemos utilizar os dois pontos ":". Onde o valor a esquerda é o valor de início e o valor da direita o de fim. O mesmo pode ser alcançado utilizando a função seq.

A função seq é mais flexível pois permite especificar valores de incrementos.

```
> c(seq(1,5,0.4))  
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6 5.0
```

A forma de acessar um elemento em um vetor é através de colchetes e o valor do índice desejado. Importante é perceber que os índices se iniciam em 1 (diferentemente de muitas linguagens de programação em que se iniciam em 0).

```
> x <- c(seq(1,5,0.4))  
> x  
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6 5.0  
> x[3]  
[1] 1.8  
> x[1]  
[1] 1  
> x[8]
```

```
[1] 3.8  
>x[1]<- 1000  
>x  
[1] 1000 0.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6 5.0
```

Um dos recursos interessantes do R é a sua flexibilidade. Para manipular/recuperar elementos de vetores é possível especificar um novo vetor como índice. Por exemplo, podemos alterar os valores de índice 5 e 7 de uma única vez. E isso também funciona para intervalos usando a função sequence (seq) ou ":".

```
>x[c(5,7)]  
[1] 2.6 3.4  
>x[c(5,7)]<- 10  
>x  
[1] 1000 0.0 1.4 1.8 2.2 10.0 3.0 10.0 3.8 4.2 4.6 5.0  
>x[c(5,7)]<- c(15, 20)  
>x  
[1] 1000 0.0 1.4 1.8 2.2 15.0 3.0 20.0 3.8 4.2 4.6 5.0
```

O R possui uma possibilidade interessante que é atribuir nomes para os elementos do vetor. Para isso utilizamos a função names. Com isso teremos apelidos para cada índice e podemos fazer referência sem especificarmos o índice real.

```
> var3a7 <- c(seq(3:7))

> names(var3a7) <- c("primeiro valor", "segundo valor", "terceiro valor", "quarto
valor", "quinto valor")

> var3a7

primeiro valor segundo valor terceiro valor quarto valor quinto valor

1           2           3           4           5

> var3a7[3]

terceiro valor

3

> var3a7["terceiro valor"]

terceiro valor

3
```

Operações sobre vetores são suportadas. Imagine somar um vetor ao outro, ou realizar uma operação entre um valor e um vetor.

```
> vetor <- c(10, 20, 30)

> vetor

[1] 10 20 30

> vetor / 2

[1] 5 10 15

> vetor * 2
```

```
[1] 20 40 60  
> vetor + 1  
[1] 11 21 31  
> vetor2 <- c(1, 2, 3)  
> vetor - vetor2  
[1] 9 18 27
```

Vetores podem conter códigos não disponíveis, mas que em geral não podem ser simplesmente ignorados. Quando uma estrutura (como um vetor) não possui um valor específico, ele é representado pelas letras NA.

## Matrizes

Matrizes é o nome que damos para vetores de mais de uma dimensão. Este tipo de estrutura nos facilita a representar diferentes problemas do dia a dia.

A função matrix nos permite criar uma matriz nova. A sua sintaxe é mostrada a seguir (você também pode usar a função help para lhe auxiliar).

matrix(<valores>, <número de linhas>, <número de colunas>)

```
> matrix(1:10,2,5)  
[1] [2] [3] [4] [,5]  
[1,] 1 3 5 7 9  
[2,] 2 4 6 8 10
```

Perceba que foi necessário passar um vetor com 10 posições, pois existem 10 slots/posições dentro de uma matriz 2 x 5.

Para fazer acesso/manipulação de um elemento em uma matriz o processo é bastante similar ao que temos na manipulação de vetores simples. A diferença é que precisamos referenciar a linha e coluna desejados (lembre-se que os índices começam em 1).

No exemplo anterior, para recuperar o valor 8 precisamos especificar a linha 2 e coluna 4. Podemos atribuir a matriz a uma variável ou podemos fazer o acesso diretamente.

```
> matriz <- matrix(1:10,2,5)  
  
> matriz[2,4]  
  
[1] 8  
  
> matrix(1:10,2,5)[2,4]  
  
[1] 8
```

Como dito anteriormente, o R possui algumas flexibilidades para facilitar o trabalho de quem o utiliza. É possível recuperar/manipular todos os valores de uma linha ou coluna de uma única vez. Perceba que os índices também podem ser intervalos.

*Podemos utilizar # para introduzir comentários ao nosso código. A sua utilização pode melhorar muito a legibilidade do código.*

```
> matriz <- matrix(1:15,3,5)  
  
> matriz  
  
[1] [2] [3] [4] [5]
```

```
[1] 1 4 7 10 13  
[2] 2 5 8 11 14  
[3] 3 6 9 12 15  
  
> matriz[,1] #primeira coluna  
  
[1] 1 2 3  
  
> matriz[,4] #quarta coluna  
  
[1] 10 11 12  
  
> matriz[2,] #segunda linha  
  
[1] 2 5 8 11 14  
  
> matriz[2:3,] #segunda e terceira linha  
  
[1] [2] [3] [4] [5]  
  
[1] 2 5 8 11 14  
[2] 3 6 9 12 15
```

## Plotando Vetores e Matrizes

Com dados armazenados dentro de estruturas do R, pode ser necessário sua visualização de forma mais interessante. O R permite diferentes formas de plotagens. Uma das mais simples é através da função barplot que gera um gráfico de barras para nós. O resultado do script a seguir está ilustrado na Figura 170.

```
> valores <- c(3,5,1)  
  
> names(valores) <- c("Categoria A", "Categoria B", "Categoria C")
```

```
> barplot(valores)
```

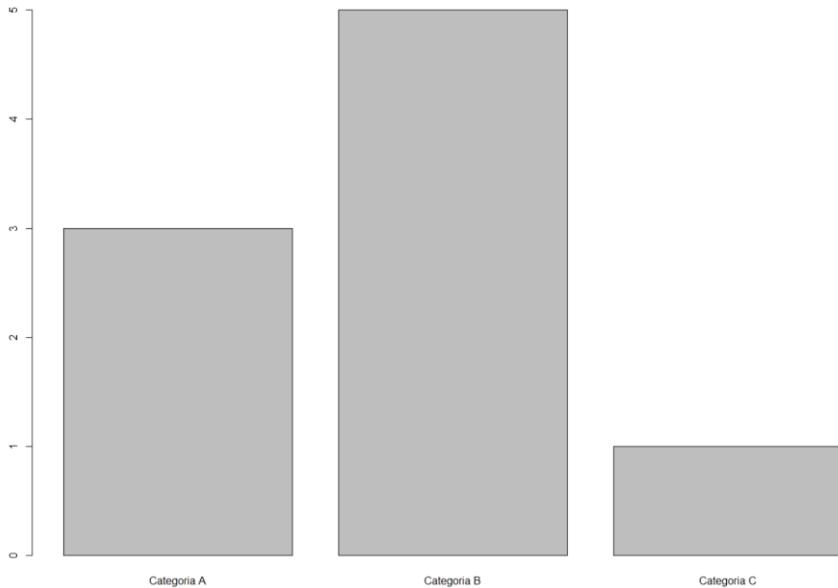


Figura 170 – Resultado da função barplot

A função `barplot` é bastante interessante para construção de gráficos simples, mas e se o desejado for uma plotagem no eixo X (horizontal) e no eixo Y (vertical)?

Podemos utilizar a função `plot` para isso. Veja o exemplo a seguir e o resultado na Figura 171.

```
> x <- c(seq(1,6,0.1))
```

```
> y <- x^2
```

```
> plot(x,y)
```

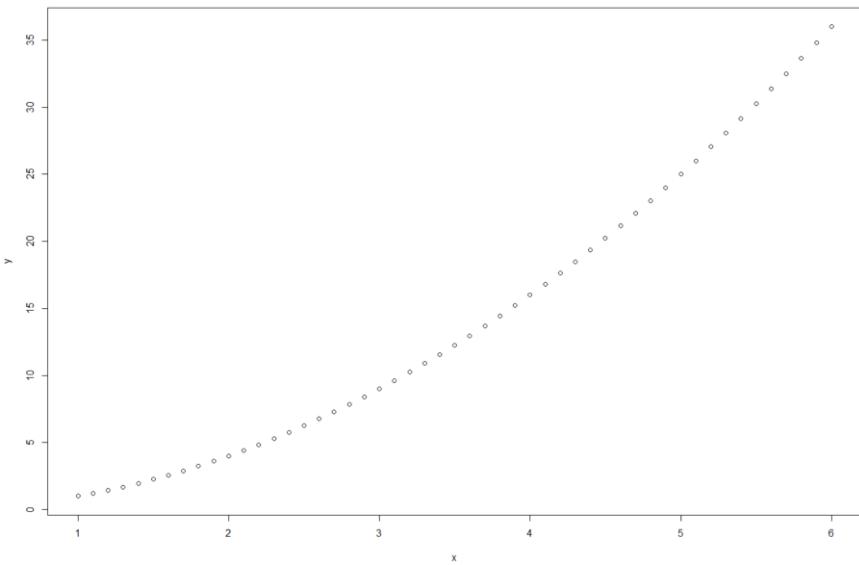


Figura 171 – Resultado da função plot

Para matrizes que são vetores de duas dimensões é possível utilizar outras funções para explorarmos os dados.

```
> area <- matrix(0, 10, 10)  
  
> area[c(3,8),c(3,9)] <- 1  
  
> area  
  
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]  
  
[1,] 0 0 0 0 0 0 0 0 0 0  
[2,] 0 0 0 0 0 0 0 0 0 0
```

```
[3,] 0 0 1 0 0 0 0 0 1 0
```

```
[4,] 0 0 0 0 0 0 0 0 0 0
```

```
[5,] 0 0 0 0 0 0 0 0 0 0
```

```
[6,] 0 0 0 0 0 0 0 0 0 0
```

```
[7,] 0 0 0 0 0 0 0 0 0 0
```

```
[8,] 0 0 1 0 0 0 0 0 1 0
```

```
[9,] 0 0 0 0 0 0 0 0 0 0
```

```
[10,] 0 0 0 0 0 0 0 0 0 0
```

Neste momento temos alguns valores 1 e o restante 0. Para ressaltar estes valores "1" podemos utilizar uma função chamada `contour`, que criará um mapa de contorno. Acompanhe a Figura 172.

```
> contour(area)
```

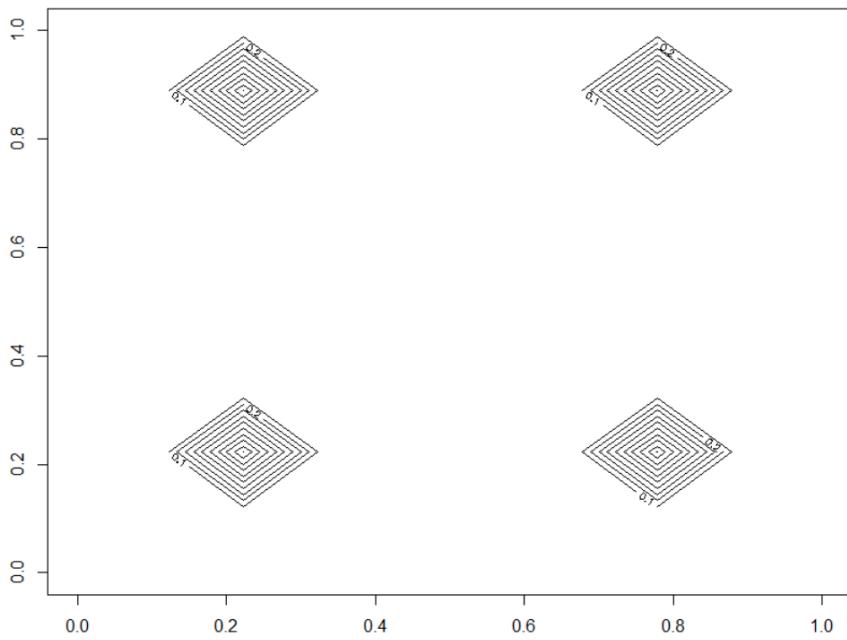


Figura 172 – Resultado da função contour

E também podemos criar uma perspectiva em 3D. O expand serve para colocar a inclinação vista na imagem. Tente por conta própria removendo o segundo parâmetro! Veja esta plotagem na Figura 173.

```
> persp(area, expand=0.3)
```

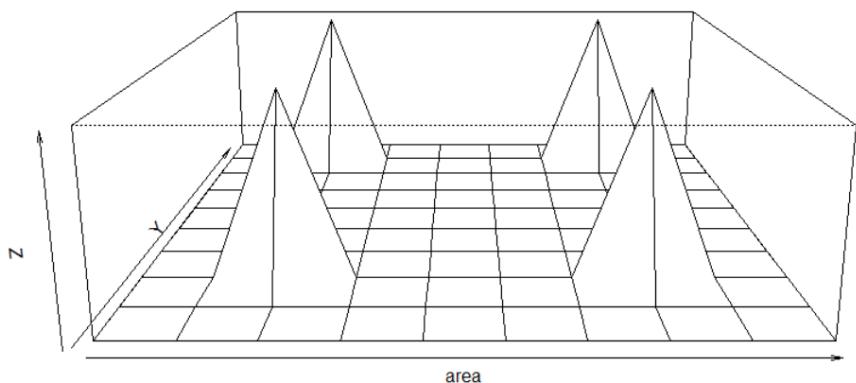


Figura 173 – Resultado da função persp

O R fornece vários conjuntos de dados de exemplo que podemos utilizar em nossos experimentos. Existe um conjunto de dados chamado *volcano*, que representa um vulcão dormente da Nova Zelândia. Este conjunto de dados é uma matriz  $87 \times 61$ .

Não vamos representar aqui o conteúdo da matriz (você pode explorá-la em seu próprio ambiente), mas podemos aplicar as funções de contorno e de perspectiva, além da função *image* capaz de gerar um mapa de calor sobre a área do vulcão. Como observado na Figura 174, 175 e 176.

```
> contour(volcano)
```

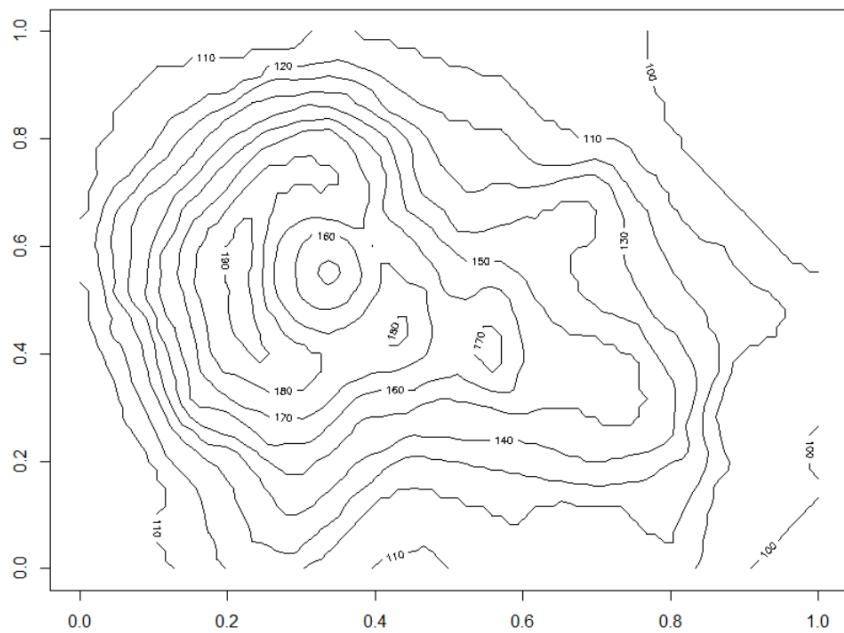


Figura 174 – Resultado da função `countour(volcano)`

```
> persp(volcano, expand=0.2)
```

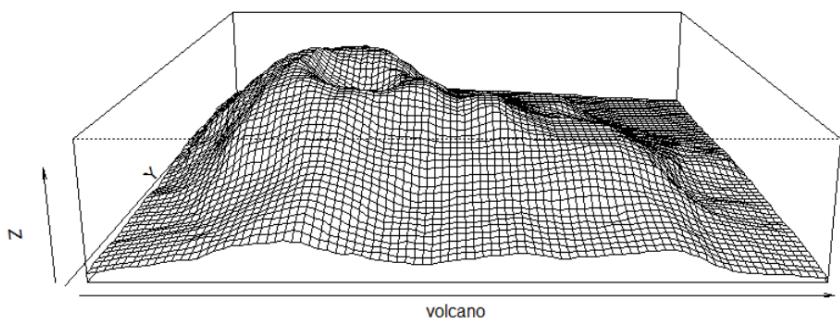


Figura 175 – Resultado da função `persp(volcano)`

```
> image(volcano)
```

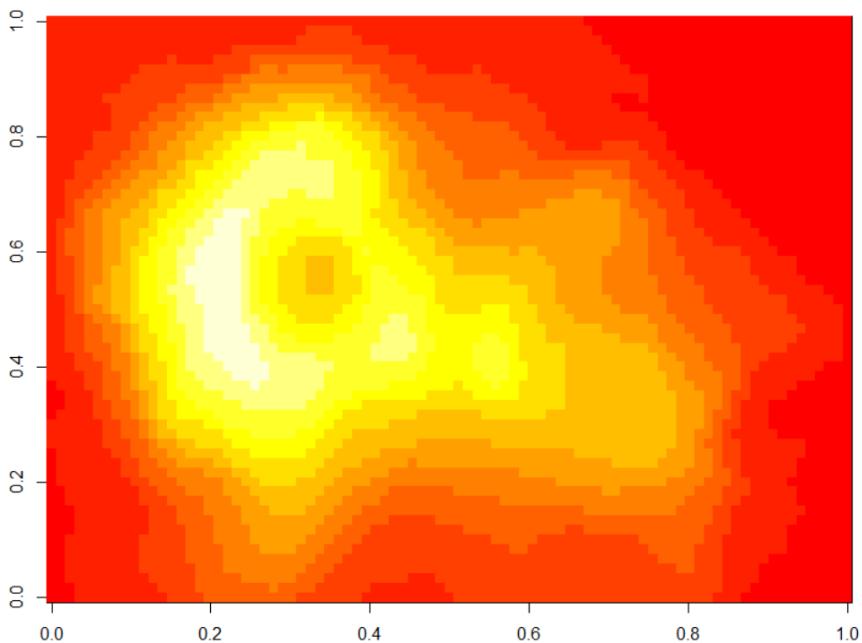


Figura 176 – Resultado da função `image(volcano)`

## Data Frames

Acabamos de analisar estruturas de dados para manipulação como vetores e matrizes. Contudo, estas estruturas sempre são elementos apartados, ou seja, uma matriz é um elemento independente de outra matriz.

Quando precisamos lidar com estruturas que são correlacionadas (e possivelmente de tipos de dados diferentes), mas de forma tal qual todas sejam tratadas como um único elemento podemos utilizar uma estrutura própria para isso: data frames.

Para efeito comparativo, você pode imaginar um data frame como sendo uma tabela em um banco de dados, na qual você possui um número determinado de colunas (cada coluna com um tipo específico de dados) e não possui um número determinado de linhas. Uma planilha no Excel poderia ser vista de forma semelhante também.

Para criar um data frame originado de estruturas diferentes podemos utilizar a função `data.frame()`. O exemplo a seguir mostra a união de três vetores em um único data frame. Perceba que não seria possível representar isso em uma matriz pois os tipos de dados de cada coluna diferem entre si.

```
> vetor1 <- c(T,T,F,T)  
  
> vetor2 <- c("Thiago", "Diego", "Maria", "Pedro")  
  
> vetor3 <- c(1:4)  
  
> df <- data.frame(vetor1, vetor2, vetor3)  
  
> print(df)  
  
vetor1 vetor2 vetor3  
  
1 TRUE Thiago 1  
2 TRUE Diego 2
```

```
3 FALSE Maria 3
```

```
4 TRUE Pedro 4
```

Para acessar dados de um data frame existem duas formas: usar duplos colchetes ou o sinal de cifrão. A notação com cifrão é a mais comumente utilizada.

```
> df[["vetor2"]]
```

```
[1] Thiago Diego Maria Pedro
```

```
Levels: Diego Maria Pedro Thiago
```

```
> df$vetor2
```

```
[1] Thiago Diego Maria Pedro
```

```
Levels: Diego Maria Pedro Thiago
```

Podemos adicionar uma coluna nova em um data frame já existente. Imagine que precisamos inserir uma nova coluna à nossa variável "df" com os sobrenomes. Para isso basta fazer atribuição de um novo vetor a uma coluna que não existe ainda. No exemplo a seguir criamos uma coluna chamada sobrenomes.

```
> df$sobrenomes <- c("Zavaschi", "Nogare", "Sobrenome A", "Sobrenome B")
```

```
> df
```

```
vetor1 vetor2 vetor3 sobrenomes
```

```
1 TRUE Thiago 1 Zavaschi
```

```
2 TRUE Diego  2 Nogare  
3 FALSE Maria 3 Sobrenome A  
4 TRUE Pedro  4 Sobrenome B
```

Muitas vezes nossos dados residem em arquivos e necessitamos utilizá-los em nossos experimentos. Para carregar dados de um arquivo CSV para dentro de um data frame podemos utilizar a função `read.csv()`.

```
> read.csv("arquivo.csv")
```

Para arquivos que não são separados por vírgulas, necessitamos utilizar a função `read.table()`. Esta função necessita de um parâmetro chamado `sep` (separador) que define qual o separador utilizado no arquivo. Imagine um arquivo separado por TAB. Um exemplo seria algo como:

```
> read.table("arquivo.txt", sep="\t")
```

Para especificar que a primeira linha são os nomes das colunas podemos utilizar o parâmetro `header`.

```
> read.table("arquivo.txt", sep="\t", header=TRUE)
```

Uma operação comum entre tabelas em bancos de dados é a operação de JOIN. No R isso não é diferente. Para obter o mesmo resultado de um JOIN no R utilizamos uma função chamada `merge()`. Por padrão o `merge` é executado nas colunas de mesmo nome.

```
> nomes <- c("Thiago", "Diego")  
> sobrenomes <- c("Zavaschi", "Nogare")
```

```
> df1 <- data.frame(nomes, sobrenomes)

> idades <- c(35, 36)

> pesos <- c(80, 150)

> df2 <- data.frame(nomes, idades, pesos)

> df1

nomes sobrenomes

1 Thiago Zavaschi

2 Diego Nogare

> df2

nomes idades pesos

1 Thiago 35 80

2 Diego 36 150

> merge(x = df1, y = df2)

nomes sobrenomes idades pesos

1 Diego Nogare 36 150

2 Thiago Zavaschi 35 80
```

Existem outras funções para se trabalhar com data frames que valem a menção:

- str(df) retorna uma descrição breve dos dados

- `names(df)` retorna o nome de cada variável
- `summary(df)` retorna algumas estatísticas básicas sobre cada variável
- `head(df)` mostra as primeiras linhas
- `tail(df)` mostra as últimas linhas

Existem diversos data frames embarcados no R que podem ser utilizados para nossos testes e exemplos. Um deles é o `mtcars` que armazena dados extraídos da revista americana "Motor Trend" de 1974 que mostra consumo e outros aspectos do projeto e performance de 32 automóveis.

*Dica: a função `help` funciona sobre os data frames do R também. Caso queira saber de mais detalhes sobre o `mtcars`: `help(mtcars)`.*

Como é um dataset grande para copiar inteiramente aqui, podemos utilizar as funções anteriores para conhecer mais sobre os dados, este é um processo comum utilizado pelo cientista de dados. Uma função que se aplica aqui também é a `rownames()`, que no caso do `mtcars` retorna os nomes das linhas (nome dos modelos).

```
> str(mtcars)
'data.frame': 32 obs. of 11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8
19.2 ...
 $ cyl  : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp   : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92
3.92 ...
 $ wt   : num  2.62 2.88 2.32 3.21 3.44 ...
```

```
$ qsec: num  16.5 17 18.6 19.4 17 ...
$ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
$ am  : num  1 1 1 0 0 0 0 0 0 0 ...
$ gear: num  4 4 4 3 3 3 3 4 4 4 ...
$ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
> names(mtcars)
```

```
[1] "mpg"   "cyl"   "disp"  "hp"    "drat"  "wt"    "qsec"  "vs"
"am"   "gear"  "carb"
```

```
> summary(mtcars)
```

	mpg	cyl	disp	hp
<i>drat</i>				
Min.	:10.40	Min.   :4.000	Min.   : 71.1	Min.   : 52.0
Min.	:2.760			
1st Qu.	:15.43	1st Qu.:4.000	1st Qu.:120.8	1st Qu.: 96.5
1st Qu.	:3.080			
Median	:19.20	Median :6.000	Median :196.3	Median :123.0
Median	:3.695			
Mean	:20.09	Mean   :6.188	Mean   :230.7	Mean   :146.7
Mean	:3.597			
3rd Qu.	:22.80	3rd Qu.:8.000	3rd Qu.:326.0	3rd Qu.:180.0
3rd Qu.	:3.920			
Max.	:33.90	Max.   :8.000	Max.   :472.0	Max.   :335.0
Max.	:4.930			

	<i>wt</i>	<i>qsec</i>	<i>vs</i>	<i>am</i>
<i>gear</i>				
Min.	:1.513	Min. :14.50	Min. :0.0000	
Min.	:0.0000	Min. :3.000		
1st Qu.	:2.581	1st Qu.:16.89	1st Qu.:0.0000	1st
Qu.	:0.0000	1st Qu.:3.000		
Median	:3.325	Median :17.71	Median :0.0000	
Median	:0.0000	Median :4.000		
Mean	:3.217	Mean :17.85	Mean :0.4375	
Mean	:0.4062	Mean :3.688		
3rd Qu.	:3.610	3rd Qu.:18.90	3rd Qu.:1.0000	3rd
Qu.	:1.0000	3rd Qu.:4.000		
Max.	:5.424	Max. :22.90	Max. :1.0000	
Max.	:1.0000	Max. :5.000		
<i>carb</i>				
Min.	:1.000			
1st Qu.	:2.000			
Median	:2.000			
Mean	:2.812			
3rd Qu.	:4.000			
Max.	:8.000			

> head(mtcars, n = 3) # o parâmetro n é opcional

	<i>mpg</i>	<i>cyl</i>	<i>disp</i>	<i>hp</i>	<i>drat</i>	<i>wt</i>	<i>qsec</i>	<i>vs</i>	<i>am</i>	<i>gear</i>	<i>carb</i>
1	21.0	6	160	110	3.90	2.88	17.8	0	1	4	4
2	21.0	6	160	110	3.90	2.88	17.8	0	1	4	4
3	22.8	4	108	95	4.08	2.32	18.0	1	1	4	1

```
Mazda RX4     21.0   6 160 110 3.90 2.620 16.46  0  1     4  
4  
Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02  0  1     4  
4  
Datsun 710    22.8   4 108  93 3.85 2.320 18.61  1  1     4  
1
```

> *tail(mtcars, n = 3)*

```
mpg cyl disp hp drat    wt qsec vs am gear carb  
Ferrari Dino 19.7   6 145 175 3.62 2.77 15.5  0  1     5     6  
Maserati Bora 15.0   8 301 335 3.54 3.57 14.6  0  1     5     8  
Volvo 142E    21.4   4 121 109 4.11 2.78 18.6  1  1     4     2
```

> *rownames(mtcars)*

```
[1] "Mazda RX4"           "Mazda RX4 Wag"        "Datsun 710"  
[4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"  
[7] "Duster 360"          "Merc 240D"          "Merc 230"  
[10] "Merc 280"            "Merc 280C"          "Merc 450SE"  
[13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetwood"  
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"  
[19] "Honda Civic"         "Toyota Corolla"       "Toyota Corona"
```

[22]	<i>"Dodge Challenger"</i>	<i>"AMC Javelin"</i>	<i>"Camaro Z28"</i>
[25]	<i>"Pontiac Firebird"</i>	<i>"Fiat X1-9"</i>	<i>"Porsche 914-2"</i>
[28]	<i>"Lotus Europa"</i>	<i>"Ford Pantera L"</i>	<i>"Ferrari Dino"</i>
[31]	<i>"Maserati Bora"</i>	<i>"Volvo 142E"</i>	

As IDEs podem auxiliar na visualização dos dados. O RStudio por exemplo possui a função View() que mostra os dados de uma forma fácil de observar.

## Instalando Novos Pacotes

Vimos na sessão anterior uma introdução à linguagem R e como manipular conjuntos de dados e estruturas. Ainda que seja possível realizar tudo isso com as bibliotecas padrão do R, existem cenários em que precisamos trabalhar com bibliotecas/pacotes desenvolvidas por outras pessoas. Como já vimos, o repositório comumente utilizado é o CRAN. Para realizar download e instalação destes pacotes, utilizamos a função install.packages().

A função install.packages() também irá instalar as dependências (caso não estejam instaladas), o que é realmente útil.

Não se preocupe em tentar descobrir se um determinado pacote está instalado ou não. Você pode deixar a instrução install.packages() em seu script que caso a biblioteca já exista (ele baixa novamente e valida se é a mesma) a execução continuará normalmente.

## Instalando e Usando um Pacote do CRAN

Um dos pacotes mais comumente utilizados é o ggplot2, para criação de representações gráficas. Este pacote não é nativo no R e precisa ser instalado para sua utilização. Para efeito de simplicidade a saída abaixo foi reduzida.

```
> install.packages("ggplot2")
```

*Installing package into 'C:/Users/zavas/Documents/R/win-library/3.2'*

*(as 'lib' is unspecified)*

*also installing the dependencies 'stringi', 'magrittr', 'colorspace', 'Rcpp', 'stringr', 'RColorBrewer', 'dichromat', 'munsell', 'labeling', 'digest', 'gtable', 'plyr', 'reshape2', 'scales'*

*trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/stringi\_1.0-1.zip'*

*Content type 'application/zip' length 14270876 bytes (13.6 MB)*

*downloaded 13.6 MB*

*...*

*trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/ggplot2\_2.1.0.zip'*

*Content type 'application/zip' length 2001264 bytes (1.9 MB)*

*downloaded 1.9 MB*

*package 'stringi' successfully unpacked and MD5 sums checked*

*package 'magrittr' successfully unpacked and MD5 sums checked*

*package 'colorspace' successfully unpacked and MD5 sums checked*

*...*

```
package 'reshape2' successfully unpacked and MD5 sums checked
```

```
package 'scales' successfully unpacked and MD5 sums checked
```

```
package 'ggplot2' successfully unpacked and MD5 sums checked
```

*The downloaded binary packages are in*

```
C:\Users\zavas\AppData\Local\Temp\RtmpaqBkne\downloaded_packages
```

Para verificar maiores informações sobre um pacote que não seja nativo, também utilizamos a função `help`. Disponível apenas após a instalação do pacote.

```
> help(package = "ggplot2")
```

Sempre que formos utilizar uma determinada biblioteca (originada de um pacote), devemos utilizar a função `library`. Para usar a biblioteca `ggplot2`, devemos realizar a seguinte instrução. Mesmo após a instalação do pacote, é necessário carregar a biblioteca para utilizar suas funções. Ao fechar o R e abrir novamente, a biblioteca deverá ser carregada novamente para memória e então suas funções estarão disponíveis para uso.

```
> library(ggplot2)
```

Existem diversas funções implementadas pela `ggplot2`, o objetivo aqui não é mostrar tudo o que é possível de realizar com ela. Caso queira conhecer mais é possível ver através da sua documentação.

Abaixo temos um exemplo da utilização da qplot() (implementada pela ggplot2) na Figura 177.

```
> library(ggplot2)

> mt <- mtcars

> mt$Marchas <- factor(mt$gear, levels=c(3,4,5), labels=c("3 marchas", "4 marchas", "5 marchas"))

> qplot(mpg, data=mt, geom="density", fill=Marchas, alpha=I(.5), main="Distribuição Baseada em Consumo", xlab="Milhas por Galão (mpg)", ylab="Densidade")
```

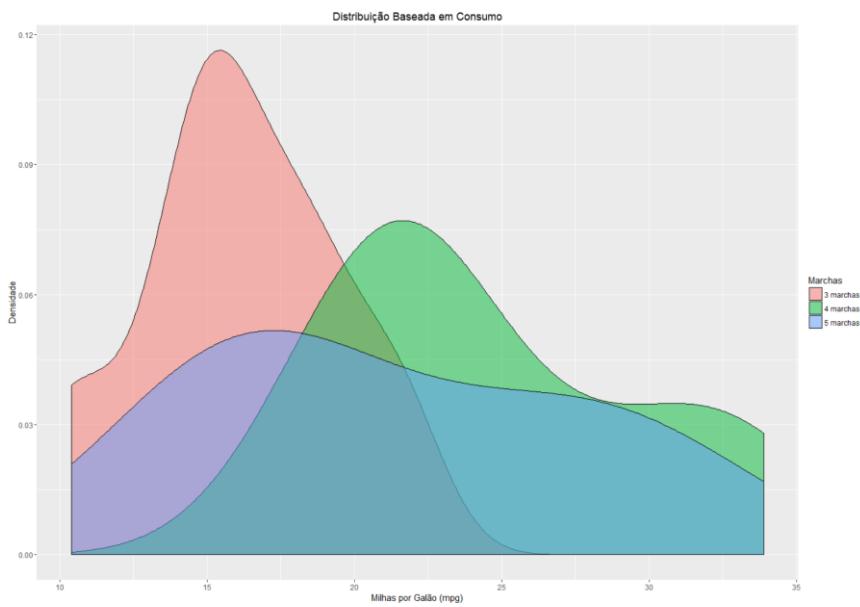


Figura 177 – Resultado da função qplot (biblioteca ggplot2)

## Exemplos utilizados no Capítulo 1 – Dia-a-Dia do Cientista de Dados

Durante o capítulo 1 (Dia-a-dia do Cientista de Dados) alguns exemplos com a linguagem R foram apresentados para ilustrar as atividades. Nos próximos parágrafos você será conduzido a reproduzir estes mesmos exemplos no seu ambiente. Após a base de entendimento da linguagem R apresentada nos parágrafos acima, praticamente toda a estrutura de linguagem, forma de pensar e escrever os códigos, sintaxe e comandos foram cobertos. Este entendimento é importante para acompanhar a criação dos elementos apresentados a seguir. Cada problema apresentado e explicado no primeiro capítulo utilizou estes códigos, e as funções que suportaram as análises serão explicadas em cada exemplo começando por Regressão Linear, seguindo para criação do Histograma e Diagrama de Caixas, e encerrando com os códigos para geração dos Clusters.

### Regressão Linear

Para trabalhar com cores específicas no R é possível utilizar as cores padrão ou então instalar um pacote externo. Um dos mais famosos, é o *RcolorBrewer*, que permite utilizar uma paleta de cores personalizada para seus gráficos. Veja o código abaixo como é feita a instalação e a chamada deste pacote.

```
> install.packages("RColorBrewer")
> library(RColorBrewer)
> cores <- brewer.pal(5, "Dark2")
```

A variável *cores* recebeu 5 variações da paleta Dark2, e estas cores serão utilizadas futuramente no momento que for interessante colocar cores no gráfico. Em seguida é criada uma massa de dados aleatória, acompanhe o código abaixo a junção de diversos aprendizados já obtidos anteriormente neste capítulo. Talvez o único código que seja novo para você é a função *runif*, que gera uma quantidade de valores aleatórios respeitando a existência de casas decimais dentro de um intervalo pré-definido. Na linha do investimento

serão gerados 200 valores com possibilidade entre 3.0 e 8.0. Já na linha do faturamento, também são gerados 200 valores com intervalo entre 1.0 e 8.0.

```
> origem <- matrix(c(0,0), nrow=200, ncol=2)
> origem <- data.frame(origem)
> colnames(origem) <- c("investimento","faturamento")
> origem$investimento <- round(matrix(runif(200, 3.0, 8.0) ,nrow=200, ncol=1)[,1],
  digits = 2)
> origem$faturamento <- round( origem$investimento + runif(200,1.0, 8.0) , digits =
  2)
> dados <- origem
```

Para fazer a plotagem do gráfico, observe o código abaixo, onde a função *plot* é chamada e alguns parâmetros são passados. A saber, o parâmetro *main* recebe o título do gráfico, o parâmetro *xlab* é a descrição do eixo X, o *ylab* é como o *xlab* porém para o eixo Y. O parâmetro *col* recebe as cores, e recorde que nas linhas de código que vimos foi gerada uma variável com 5 cores, ela é usada neste momento. O parâmetro *pch* identifica qual será o símbolo utilizado na plotagem.

```
> plot(dados,col=cores[1], pch=20
  , main="Investimento X Faturamento"
  , xlab="Investimento em Marketing (vezes 10)"
  , ylab="Faturamento em R$ (vezes 100)")
```

Veja como fica o gráfico gerado a partir deste código na Figura 178.

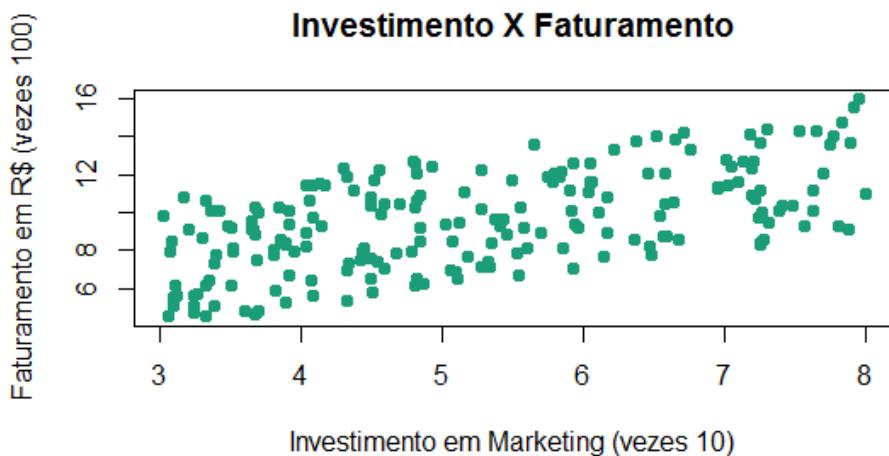


Figura 178 – Gráfico resultado da função *plot*. Investimento x Faturamento

A regressão linear é criada a partir da função *lm*, que significa *linear model*, e recebeu como parâmetro as duas colunas do data frame existente.

```
> Regressao = lm( dados$faturamento ~ dados$investimento )
```

Em seguida uma linha é traçada a partir da chamada da função *abline* e imprime esta linha no gráfico já existente.

```
> abline(Regressao, col=cores[2])
```

Veja como fica o gráfico com a linha de regressão na Figura 179.

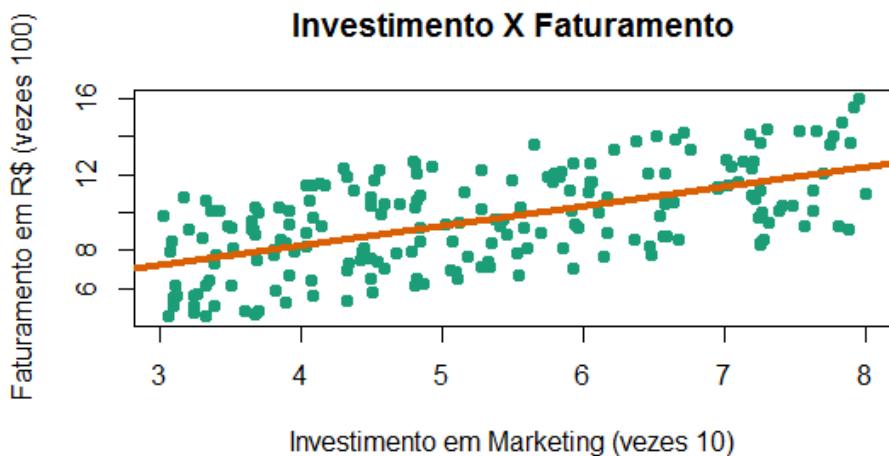


Figura 179 – Aplicação da função abline

Por fim, para calcular o valor da variável dependente, que está no eixo Y a partir dos dados calculados na regressão linear, utilize a formula abaixo.

$$y = a + bx$$

Em linguagem R, pode ser representada desta forma:

```
> investimento = 5  
> Y = Regressao$coefficients[1] + investimento * Regressao$coefficients[2]
```

Onde a variável investimento é o valor que existe no eixo X e será usado como ponto para calcular o ponto da reta que toca neste valor.

Para colocar um ponto na reta que representa o valor onde este investimento foi feito, utilize o seguinte código.

```
> points(investimento, Y, col=cores[3], lwd=4, pch=20)
```

Para traçar uma reta na vertical do ponto definido no eixo X até a linha da regressão, e em seguida para ir deste ponto da reta até o valor do eixo Y, acompanhe como fazer.

```
> lines(investimento, Y, type="h", col=cores[3])
> segments(0, Y, investimento, Y, col=cores[3])
```

Veja como fica o gráfico a partir destas novas linhas na Figura 180.

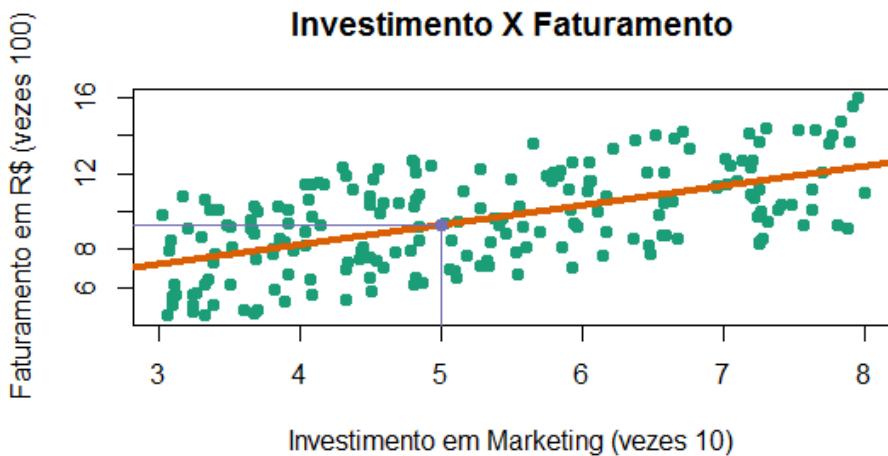


Figura 180 – Gráfico com a aplicação das funções *lines* e *segments*

O valor do eixo Y, a partir do valor 5 no eixo X é de 9.30. Este valor pode variar no seu exemplo, uma vez que os dados destes pontos são aleatórios e foram gerados a partir da função *runif*.

## Histograma e Diagrama de Caixa

Seguindo a mesma linha do exemplo anterior, da Regressão Linear, neste exemplo é chamado o pacote *RColorBrewer* para mudança das cores. Se você fez o exercício anterior não é necessário instalar o pacote, basta carregar através da função library, como no código a seguir.

```
> library(RColorBrewer)
> cores <- brewer.pal(5, "Dark2")
```

Para gerar a massa de dados que será plotada no histograma, deve-se criar um Data Frame vazio para em seguida popular com dados de exemplo. Veja como criar este Data Frame vazio chamado Origem:

```
> origem <- matrix(c(1,1), nrow=1, ncol=2)
> origem <- data.frame(origem)
> colnames(origem) <- c("categoria", "preco")
> origem$preco <- sample(0:100, 1, replace = TRUE)
```

Então o passo seguinte é colocar dados no Data Frame, serão cinco grupos de dados aleatórios, cada um com seus respectivos valores, sendo os intervalos entre 0 e 100, 101 e 200, 201 e 300, 301 e 400 e finalizando com 401 e 500.

```
> #Categoria 1
> aux <- matrix(c(1, nrow=4, ncol=2)
> aux <- data.frame(aux)
> colnames(aux) <- c("categoria", "preco")
> aux$preco <- sample(0:100, 4)
> origem <- rbind(origem, aux)
```

```
> #Categoria 2  
> aux <- matrix(c(2), nrow=10, ncol=2)  
> aux <- data.frame(aux)  
> colnames(aux) <- c("categoria", "preco")  
> aux$preco <- sample(101:200, 10)  
> origem <- rbind(origem, aux)  
  
> #Categoria 3  
> aux <- matrix(c(3), nrow=20, ncol=2)  
> aux <- data.frame(aux)  
  
> colnames(aux) <- c("categoria", "preco")  
> aux$preco <- sample(201:300, 20)  
> origem <- rbind(origem, aux)  
  
> #Categoria 4  
> aux <- matrix(c(4), nrow=10, ncol=2)  
> aux <- data.frame(aux)  
  
> colnames(aux) <- c("categoria", "preco")  
> aux$preco <- sample(301:400, 10)  
> origem <- rbind(origem, aux)  
  
> #Categoria 5  
> aux <- matrix(c(5), nrow=5, ncol=2)  
> aux <- data.frame(aux)  
> colnames(aux) <- c("categoria", "preco")  
> aux$preco <- sample(401:500, 5)  
> origem <- rbind(origem, aux)  
  
> dados <- origem
```

Por fim, chama-se a função *hist* para criar o histograma. Todos os códigos utilizados neste exemplo já foram explicados anteriormente neste capítulo.

```
> hist(dados$preco, breaks=5, main="Frequência por Faixa de preços"  
, xlab="Faixa de Preço"
```

```
, ylab = "Frequência"  
, col = cores)
```

O histograma impresso deve ser conforme a Figura 181.

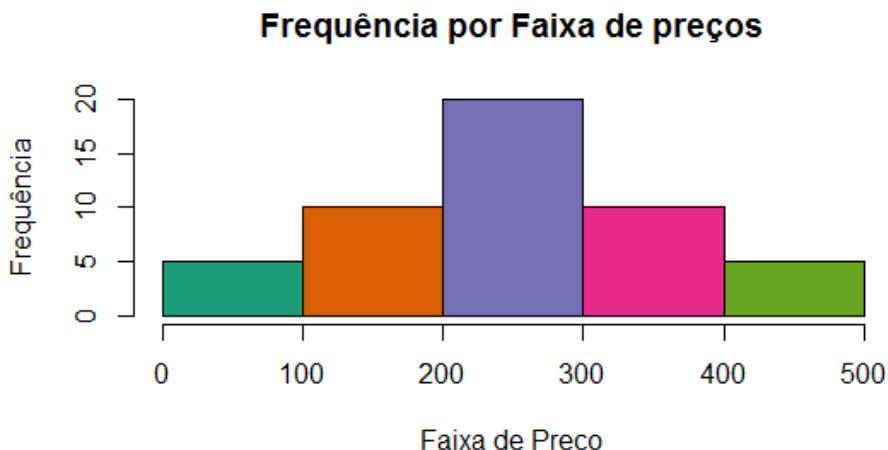


Figura 181 – Histograma das faixas de preços

Uma outra possibilidade de plotagem com estes mesmos dados criados é o uso de Diagrama de Caixas. Para entender este diagrama, vamos isolar a categoria 3, que representa os produtos de valor entre 201 e 300, para que seja mais fácil a explicação deste diagrama. Para fazer um filtro do Data Frame e atribuir à um novo, pode-se usar a função *subset* passando como primeiro parâmetro o Data Frame original e como segundo parâmetro o filtro a ser aplicado, veja como fica este código.

```
> categoria3 <- subset(dados, categoria == 3)
```

Para gerar o diagrama de caixa após a criação do Data Frame categoria3, é necessário chamar a função *boxplot* que é responsável por criar este diagrama. Esta função recebe como parâmetros o Data Frame, as colunas (ou variáveis) que serão apresentadas, a etiqueta para os eixos X e Y e também o vetor de cores. O código abaixo é responsável por plotar o diagrama de caixa baseado no Data Frame categoria3.

```
> boxplot(data = categoria3, preco ~ categoria  
, xlab="Categorias"  
, ylab="Preço", main="Diagrama de Caixa"  
, col=cores[3])
```

O resultado desta plotagem pode ser visto na Figura 182.

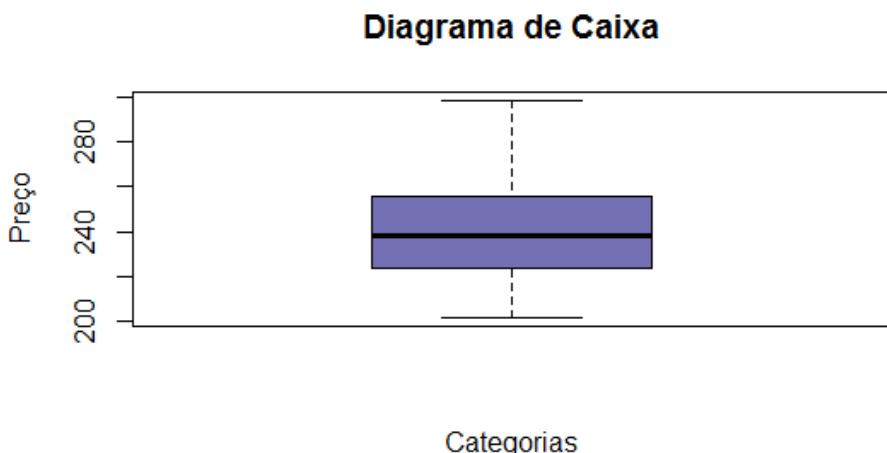


Figura 182 – Resultado da função *boxplot*

Para avançar a leitura para a base de dados completa, basta alterar o *data frame* que gerou o diagrama, mudando de categoria3 para dados, que é

o Data Frame que possui todos os dados gerados aleatoriamente no começo deste exemplo. Veja como fica a seguir.

```
> boxplot(data = dados, preco ~ categoria, xlab="Categorias"  
, ylab="Preço", main="Intervalos de Preço por Categoria"  
, col=cores)
```

E por fim, o código gerou o diagrama das cinco categorias, ilustrado na Figura 183.

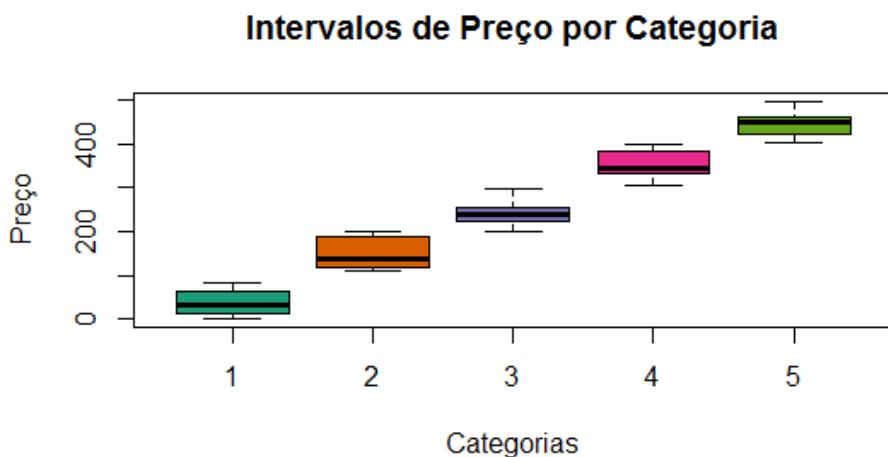


Figura 183 – Aplicação da função boxplot

## Clusterização

Seguindo o mesmo início de código dos dois exemplos mostrados anteriormente, para garantir as cores desejadas carrega-se o pacote *RcolorBrewer* através do código a seguir.

```
> library(RColorBrewer)  
  
> cores <- brewer.pal(10, "Paired")
```

O primeiro exemplo utiliza a base de dados do exemplo criado na explicação do Histograma, mas segue novamente o código, caso queira acompanhar:

```
> origem <- matrix(c(1.0,1.0), nrow=1, ncol=2)  
  
> origem <- data.frame(origem)  
  
> colnames(origem) <- c("categoria", "preco")  
  
> origem$preco <- runif(0:100, 1, replace = TRUE)  
  
> origem$categoria <- runif(0:100, 1, replace = TRUE)  
  
#  
  
> #Categoria 1  
  
> aux <- matrix(c(1), nrow=20, ncol=2)  
  
> aux <- data.frame(aux)  
  
> colnames(aux) <- c("categoria", "preco")  
  
> aux$preco <- sample(0:100, 20)  
  
> aux$categoria <- runif(20, 0.01, 1.0)  
  
#  
  
> origem <- rbind(origem,aux)
```

```
> #Categoria 2  
  
> aux <- matrix(c(2), nrow=20, ncol=2)  
  
> aux <- data.frame(aux)  
  
> colnames(aux) <- c("categoria", "preco")  
  
> aux$preco <- sample(101:200, 20)  
  
> aux$categoria <- runif(20, 1.01, 2.0)  
  
  
  
> origem <- rbind(origem, aux)  
  
  
  
> #Categoria 3  
  
> aux <- matrix(c(3), nrow=20, ncol=2)  
  
> aux <- data.frame(aux)  
  
> colnames(aux) <- c("categoria", "preco")  
  
> aux$preco <- sample(201:300, 20)  
  
> aux$categoria <- runif(20, 2.01, 3.0)  
  
  
  
> origem <- rbind(origem, aux)  
  
  
  
> #Categoria 4
```

```
> aux <- matrix(c(4), nrow=20, ncol=2)

> aux <- data.frame(aux)

> colnames(aux) <- c("categoria", "preco")

> aux$preco <- sample(301:400, 20)

> aux$categoria <- runif(20, 3.01, 4.0)

> origem <- rbind(origem, aux)

> #Categoria 5

> aux <- matrix(c(5), nrow=20, ncol=2)

> aux <- data.frame(aux)

> colnames(aux) <- c("categoria", "preco")

> aux$preco <- sample(401:500, 20)

> aux$categoria <- runif(20, 4.01, 5.0)

> origem <- rbind(origem, aux)

> dados <- origem
```

A diferença é que neste momento, deve-se criar uma representação visual através da chamada da função *plot* e não da função *hist*, como foi feito anteriormente. A função *plot* neste caso será toda impressa em preto, para depois poder aplicar os grupos que farão a separação dos valores. Veja o código que gera os pontos de impressão.

```
> plot(dados, pch=19, main="Preço de Produtos", ylab="Preço",  
xlab="Índice de Vendas")
```

O gráfico impresso segue a representação visual a seguir , na Figura 184.

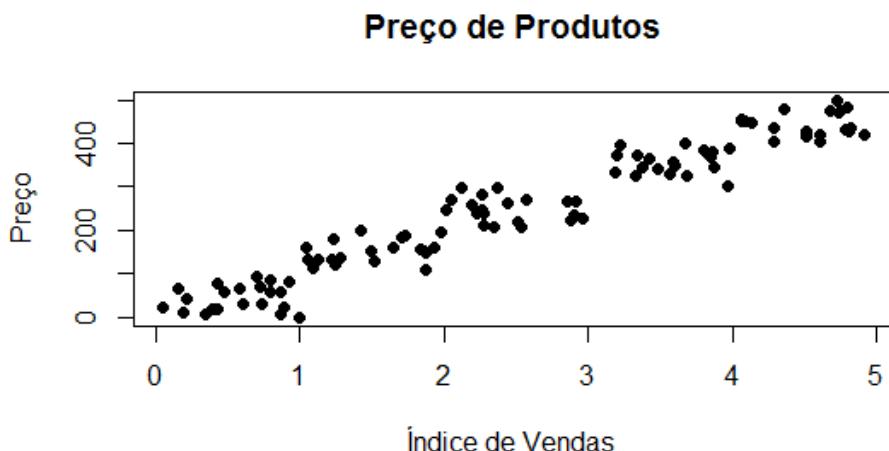


Figura 184 – Aplicação da função *plot* de acordo com o exemplo anterior

A função responsável por separar os dados em determinados grupos é a função *KMeans*, que recebe por parâmetro dois valores, sendo o primeiro o data frame que possui os dados e o segundo a quantidade de "K"s, ou grupos, que devem ser separados. Esta função retorna alguns elementos, que

através da atribuição é armazenado em uma outra variável, chamada grupos. Acompanhe a chamada desta função a seguir.

```
> grupos <- kmeans(dados,5)
```

Após essa atribuição dos valores agrupados em cinco segmentos para a variável grupos, é possível recuperar visualmente esta representação utilizando novamente a função *plot*. Porém, desta vez as cores são atribuídas para o valor do vetor existente no método grupos\$cluster. Este método possui qual é o grupo atribuído para o elemento daquela posição do vetor, assim, quando este elemento for plotado no gráfico e respeitar esta configuração de cor, todos elementos daquele grupo atribuído terão as mesmas cores na representação visual. Acompanhe como é esta chamada.

```
> plot(dados, col=cores[grupos$cluster], pch=19,  
main="Preço de Produtos", ylab="Preço", xlab="Índice de Vendas")
```

E o gráfico com as cores separadas fica conforme demonstrado na Figura 185.

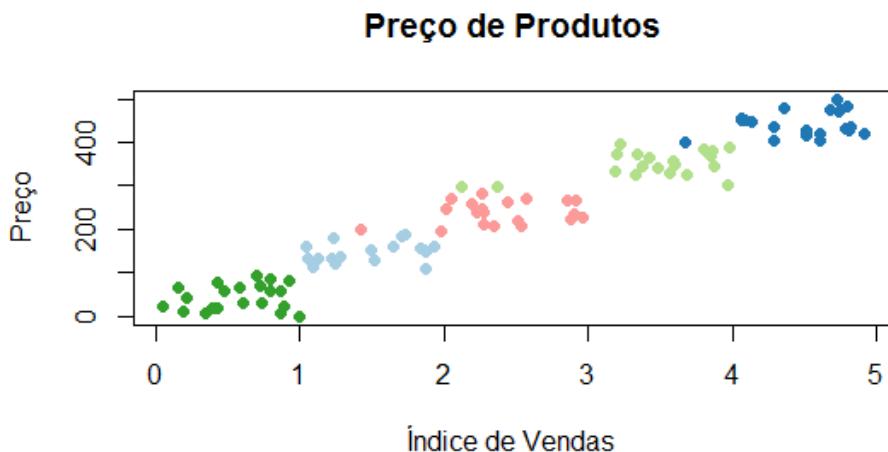


Figura 185 – Grupos evidenciados com a segmentação de cores

Para colocar nesta representação visual também os pontos centrais de cada cluster, ou grupo, a chamada à função *points* é feita passando por parâmetro o local de cada ponto, o símbolo que será utilizado e a cor. O valor de cada ponto é obtido através do método *grupos\$center*, que possui o valor de cada centroide que foi separado pelo algoritmo através da chamada da função *KMeans*. O código para colocar os pontos no gráfico está representado abaixo.

```
> points(grupos$centers, pch=8, cex=2)
```

E o gráfico com os pontos fica conforme mostrado na Figura 186.

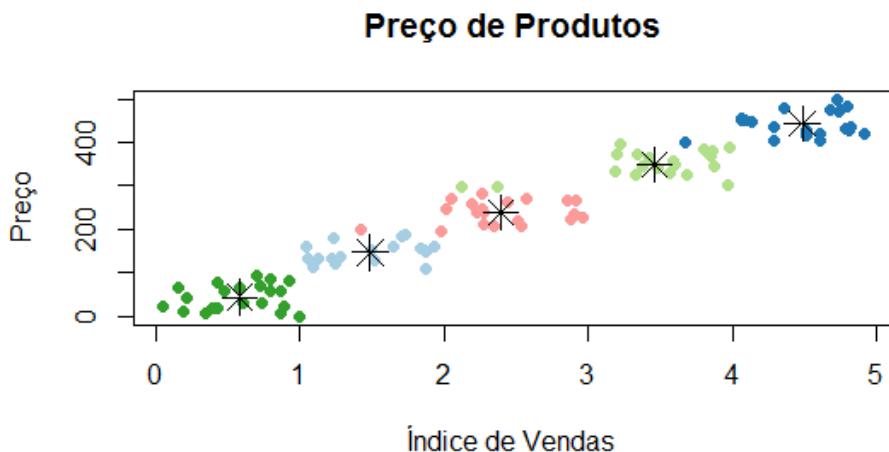


Figura 186 - Grupos evidenciados com a segmentação de cores e representação dos centroides

Continuando a pensar nos aglomerados que são calculados utilizando a função *KMeans*, outro exemplo a ser visto é com a massa de dados utilizada no exemplo da Regressão Linear. Caso não tenha mais o data frame, ele pode ser gerado da seguinte forma.

```
> origem2 <- matrix(c(0,0), nrow=200, ncol=2)
> origem2 <- data.frame(origem2)
> colnames(origem2)<- c("investimento","faturamento")
> origem2$investimento <- round(matrix(runif(200, 3.0, 8.0),nrow=200, ncol=1)[,1], digits = 2)
> origem2$faturamento <- round( origem2$investimento + runif(200,1.0, 8.0) , digits = 2)
> dados2 <- origem2
```

Então um gráfico com os pontos é criado com o uso da função *plot*.

```
> plot(dados2, pch=19  
, main="Investimento X Faturamento"  
, xlab="Investimento em Marketing (vezes 10)"  
, ylab="Faturamento em R$ (vezes 100)")
```

E o gráfico fica visualmente fica conforme mostrado na Figura 187.



Figura 187 – Gráfico com os dados do exemplo plotados

Ao separar estes valores em 10 conglomerados (ou grupos) é possível acompanhar que alguns grupos de investimentos tiveram retornos melhores que outros da mesma faixa de preço. Separe estes dados em 10 grupos, como foi feito através do seguinte código.

```
> grupos2 <- kmeans(dados2,10)
```

Na sequência é preciso chamar a função *plot* novamente passando os valores do método *grupos\$cluster* como definição para o parâmetro *col*, conforme o código abaixo.

```
> plot(dados2, col=cores[grupos2$cluster], pch=19  
, main="Investimento X Faturamento"  
, xlab="Investimento em Marketing (vezes 10)"  
, ylab="Faturamento em R$ (vezes 100)")
```

Aproveite e coloque os pontos de cada centroide utilizando a função *points* e seus parâmetros:

```
> points(grupos2$centers, pch=8, cex=2)
```

O gráfico final fica conforme a Figura 188, já com os centroides apresentados:

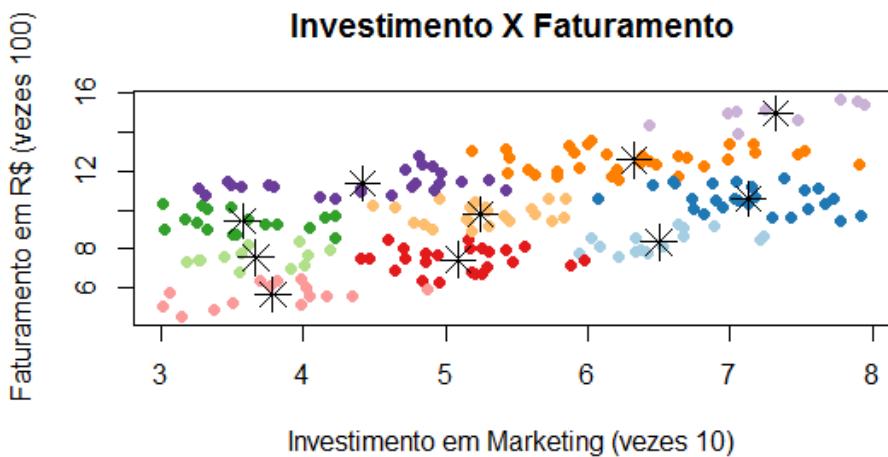


Figura 188 – Centroides e grupos evidenciados

Com isso todos os exemplos do capítulo inicial do livro foram cobertos e você poderá replicar estes exemplos em seu ambiente. Lembrando que, como se tratam de valores gerados aleatoriamente, seus gráficos ficarão ligeiramente diferentes dos apresentados aqui, mas ainda assim seguirão uma estrutura muito semelhante.

## Azure Machine Learning com R

Depois desta introdução ao R, vejamos como integrar os nossos conhecimentos em R com a plataforma do Azure Machine Learning.

Existem dois módulos de R atualmente na plataforma: **Create R Model** e **Execute R Script**.

Vamos inicialmente elencar algumas das possibilidades de utilização do R dentro da plataforma, e na sequência vamos mostrar como cada cenário pode ser implementado.

Os cenários #1 a #6 cobrem a utilização do módulo Execute R Script e o cenário #7 cobre a utilização do módulo Create R Model.

### O Módulo “Execute R Script”

Este módulo possui três inputs e dois outputs, conforme apresentado na Figura 189.



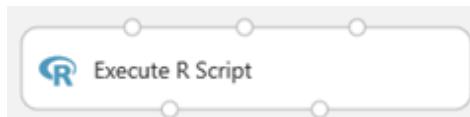


Figura 189 – Módulo Execute R Script

Sendo datasets os dois primeiros inputs, o terceiro é um script bundle (zip) – veremos mais sobre nos cenários a seguir.

A primeira saída deste módulo é um *dataset* (tradicional do *AzureML*), e o segundo output é um dataset também, mas é do R, chamada de *RDevice* (imagine que seu script em R gerou uma imagem, é possível consultar esta imagem através deste output, entre outras funções).

### Utilizar um script R codificado diretamente no componente

Ao selecionar o módulo, na janela de propriedades há uma propriedade chamada R Script e um script de exemplo. Este é o código que será executado quando o módulo for executado, acompanhe a Figura 190.

## Properties

### ▲ Execute R Script

R Script

```

1 # Map 1-based optional input ports to variables
2 dataset1 <- maml.mapInputPort(1) # class: data.frame
3 dataset2 <- maml.mapInputPort(2) # class: data.frame
4
5 # Contents of optional Zip port are in ./src/
6 # source("src/yourfile.R");
7 # load("src/yourData.rdata");
8
9 # Sample operation
10 data.set = rbind(dataset1, dataset2);
11
12 # You'll see this output in the R Device port.
13 # It'll have your stdout, stderr and PNG graphics device(s).
14 plot(data.set);
15
16 # Select data.frame to be sent to the output Dataset port
17 maml.mapOutputPort("data.set");

```

Figura 190 – Script de exemplo presente no módulo Execute R Script

Este código pode ser praticamente qualquer código em R, recomendamos que o código seja testado em um ambiente menor/controlado anteriormente (RStudio, RTVS, etc.).

Lembre-se que esse módulo é executado de forma modularizada (*sandbox*) e o código não possui conectividade com a internet. Isso significa que não é possível se comunicar diretamente ao CRAN para instalar novos pacotes. Veja nos cenários a seguir sobre como utilizar uma biblioteca externa que não esteja instalada no AzureML.

## Utilização de um script codificado fora da plataforma

Nem todo código a ser utilizado na plataforma necessita ser codificado nesta janela. Ele pode ser oriundo de um script já codificado.

Para utilizar este script (possivelmente mais de um), você deve primeiramente compactá-lo em um arquivo .zip e fazer o upload como se fosse um upload de um dataset regular.

Neste exemplo vamos utilizar o seguinte script em R:

```
descreveDataset <- function(x) { summary(x) }
```

Basicamente será criado uma função que receberá um *data frame* de entrada e executará um *summary* deste data frame. É claro que é possível codificar isso no componente diretamente (como visto no cenário anterior), mas a ideia deste cenário é a reutilização de código já codificado/existente.

Na Figura 191 podemos observar o uso da função.

```
1 descreveDataset <- function(x) { summary(x) }
2
3 resultado <- descreveDataset(mtcars)
4
5 print(resultado)
6
6:1 [Top Level] ▾

Console C:/Users/thiagoz/AppData/Local/Temp/Temp1_scriptR.zip/ ↵
> descreveDataset <- function(x) { summary(x) }
> resultado <- descreveDataset(mtcars)
> print(resultado)
      mpg        cyl       disp        hp       drat
Min. :10.40  Min. :4.000  Min. : 71.1  Min. :52.0  Min. :2.760
1st Qu.:15.43 1st Qu.:4.000  1st Qu.:120.8  1st Qu.:96.5  1st Qu.:3.080
Median :19.20  Median :6.000  Median :196.3  Median :123.0  Median :3.695
Mean   :20.09  Mean   :6.188  Mean   :230.7  Mean   :146.7  Mean   :3.597
3rd Qu.:22.80 3rd Qu.:8.000  3rd Qu.:326.0  3rd Qu.:180.0  3rd Qu.:3.920
Max.   :33.90  Max.   :8.000  Max.   :472.0  Max.   :335.0  Max.   :4.930
      wt         qsec        vs         am         gear
Min. :1.513  Min. :14.50  Min. :0.0000  Min. :0.0000  Min. :3.000
1st Qu.:2.581 1st Qu.:16.89  1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:3.000
Median :3.325  Median :17.71  Median :0.0000  Median :0.0000  Median :4.000
Mean   :3.217  Mean   :17.85  Mean   :0.4375  Mean   :0.4062  Mean   :3.688
3rd Qu.:3.610 3rd Qu.:18.90  3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.:4.000
Max.   :5.424  Max.   :22.90  Max.   :1.0000  Max.   :1.0000  Max.   :5.000
      carb
Min. :1.000
1st Qu.:2.000
Median :2.000
Mean   :2.812
3rd Qu.:4.000
Max.   :8.000
> |
```

Figura 191 – Uso da função `descreveDataset`

Para usar no AzureML, primeiramente crie um arquivo chamado `meuScriptR.R` com a linha de código que mostramos acima. Acompanhe isso na Figura 192.

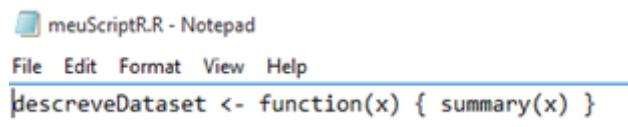


Figura 192 – Script dentro do arquivo `meuScriptR.R`

Agora compacte o arquivo (para o formato .zip) e nomeie meuScriptR.zip.

Faça upload do arquivo zip para o AzureML, conforme mostrado na Figura 193.

The screenshot shows the 'Upload a new dataset' interface. At the top right is a close button (X). Below it is the title 'Upload a new dataset'. Underneath is a section labeled 'SELECT THE DATA TO UPLOAD:' with a text input field containing 'C:\Users\thiagoz\Desktop\R Blog\meuScriptR' and a 'Browse...' button. There is also a checkbox labeled 'This is the new version of an existing dataset'. The next section is 'ENTER A NAME FOR THE NEW DATASET:' with an input field containing 'meuScriptR.zip'. Below that is 'SELECT A TYPE FOR THE NEW DATASET:' with a dropdown menu showing 'Zip File (.zip)'. The final section is 'PROVIDE AN OPTIONAL DESCRIPTION:' with an input field containing 'Script em R para a função summarize.'. In the bottom right corner of the form area, there is a circular icon with a checkmark inside.

Figura 193 – Upload do script para a plataforma AzureML

Agora adicione um módulo *Execute R Script*. Localize também (nos datasets) o “meuScriptR.zip” (ou o nome que você deu) e conecte no terceiro input do módulo. Ficará algo como demonstrado na Figura 194.



Figura 194 – Experimento criado para utilizar o meuScriptR.R

Substitua o código do *Execute R Script* pelo seguinte código (para fazer chamada na função criada e armazenada no script), conforme apresentado na Figura 195.

Todos os scripts são descompactados em um diretório chamado "src".

```
source("src/meuScriptR.R")
print(descreveDataset(mtcars))
```

## Properties

### ▲ Execute R Script

#### R Script

```
1 source("src/meuScriptR.R")
2 print(descreveDataset(mtcars))
```

Figura 195 – Uso da função `descreveDataset`, dentro do AzureML

E por fim, execute o experimento. Após a execução é possível (nas propriedades do módulo de R) ir em “View output log”, que mostra o resultado da aplicação da função. Acompanhe a Figura 196.

```
[ModuleOutput]
[ModuleOutput] Loading objects:
[ModuleOutput]
[ModuleOutput] [ModuleOutput] mpg cyl disp hp
[ModuleOutput] [ModuleOutput] Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0
[ModuleOutput] [ModuleOutput] 1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5
[ModuleOutput] [ModuleOutput] Median :19.20 Median :6.000 Median :196.3 Median :123.0
[ModuleOutput] [ModuleOutput] Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7
[ModuleOutput] [ModuleOutput] 3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0
[ModuleOutput] [ModuleOutput] Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0
[ModuleOutput]
[ModuleOutput] drat wt qsec vs
[ModuleOutput] Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000
[ModuleOutput] 1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000
[ModuleOutput] Median :3.695 Median :3.325 Median :17.71 Median :0.0000
[ModuleOutput] Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375
```

Figura 196 – Saída da execução da função `descreveDataset`

## Utilizando uma biblioteca externa

Para realizar esta tarefa, os passos são muito similares ao cenário anterior. É possível fazer upload de um arquivo .zip com a biblioteca e utilizar ela nos seus experimentos. Lembrem-se que muitas bibliotecas já se encontram disponíveis no R do AzureML (leia mais sobre isso na sequência).

O ponto importante aqui é que será necessária a instalação da biblioteca. Neste exemplo estou mostrando a ggdendro. Ficaria algo como o seguinte código, apresentado na Figura 197.

```
install.packages("src/ggdendro_0.1-14.zip", lib = ".", repos =
NULL, verbose = TRUE);
library("ggdendro", lib.loc = ".", logical.return = TRUE,
```

```
verbose = TRUE);  
library(ggdendro)
```



Figura 197 – Utilização de uma biblioteca externa

## Interagindo com datasets externos

Para execução deste cenário, vamos necessitar de um arquivo .RData, e para utilizá-lo no AzureML precisaremos fazer o upload dele assim como fizemos nos cenários anteriores (em um arquivo zip).

Dentro do módulo *Execute R Package* devemos utilizar a função *load* para recuperar o conteúdo do arquivo .RData.

É possível também utilizar este dataset dentro do próprio AzureML, sem necessariamente ser no script R.

Para interagir com datasets externos (ler os inputs de entrada) e gerar um dataset que possa ser lido a partir do output de dataset (primeiro output do componente) existem algumas funções auxiliares: *maml.mapInputPort* e *maml.mapOutputPort* (MAML é um acrônimo para Microsoft Azure Machine Learning).

Então, para ler o arquivo .RData no arquivo que fizemos o upload e depois disponibilizar através do output do componente, devemos executar as seguintes instruções conforme as Figuras 198 e 199.

```
load("src/Boston.rdata")
maml.mapOutputPort("Boston");
```

### R Script

```
1 load("src/Boston.rdata")
2 maml.mapOutputPort("Boston");
```

Figura 198 – Lendo dados de um arquivo .Rdata

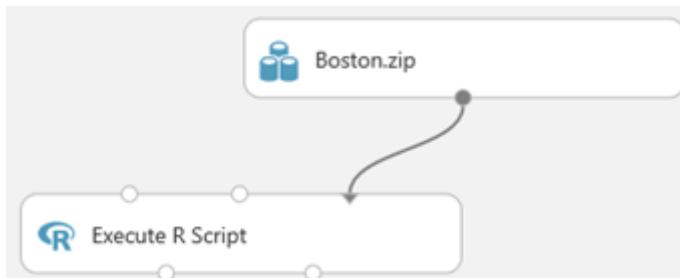


Figura 199 – Experimento para leitura de um arquivo RData

Para ler de um dataset e utilizar ele dentro do script em R, devemos utilizar um script semelhante ao mostrado a seguir. O parâmetro é a referência a qual dos dois primeiros inputs se deseja pegar o dataset.

```
dataset1 <- maml.mapInputPort(1)
dataset2 <- maml.mapInputPort(2)
```

Após a execução destes comandos, o dataset (oriundo do fluxo do AzureML) passa a ser reconhecido como um data frame dentro do script R.

## Criando um modelo de aprendizagem de máquina dentro do módulo Execute R Script.

Se existir algum algoritmo de machine learning não disponível no AzureML que você queira utilizar, é possível criar o modelo inteiro através de código em R. Acompanhe a Figura 200.

É possível utilizar o R da mesma forma que se usaria fora do AzureML. A maior vantagem desta abordagem é a possibilidade de utilizar bibliotecas e modelos que possam não estar disponíveis na plataforma ainda.

A seguir podemos ver um exemplo simples. Com o que vimos anteriormente, podemos integrar dados externos (maml.mapInputPort) dentro do R, e depois exportá-los para serem expostos para outros módulos (por exemplo o Web Service output).

```
Boston <- maml.mapInputPort(1)
install.packages("src/ggdendro_0.1-14.zip", lib = ".", repos = NULL, verbose = TRUE);
library("ggdendro", lib.loc = ".", logical.return = TRUE, verbose = TRUE);
library(ggplot2)
library(ggdendro)
library(rpart)

fit <- rpart(medv~, data=Boston)
fitr <- dendro_data(fit)
p <- ggplot() + geom_segment(data=fitr$segments, aes(x=x, y=y, xend=xend,
yend=yend)) + geom_text(data=fitr$labels, aes(x=x, y=y, label=label), size=3,
vjust=0) + geom_text(data=fitr$leaf_labels, aes(x=x, y=y, label=label), size=3,
vjust=1) + theme_dendro()

print(p)
```

```
1 Boston <- maml.mapInputPort(1)
2 install.packages("src/ggdendro_0.1-14.zip", lib = ".", repos = NULL, verbose = TRUE);
3 library("ggdendro", lib.loc = ".", logical.return = TRUE, verbose = TRUE);
4 library(ggplot2)
5 library(ggdendro)
6 library(rpart)
7 fit <- rpart(medv~, data=Boston)
8 fitr <- dendro_data(fit)
9 p <- ggplot() + geom_segment(data=fitr$segments, aes(x=x, y=y, xend=xend, yend=yend)) + geom_text(data=fitr$labels, aes(x=x, y=y, label=label),
10   size=3, vjust=0) + geom_text(data=fitr$leaf_labels, aes(x=x, y=y, label=label), size=3, vjust=1) + theme_dendro()
11 print(p)
12
```

Figura 200 – Exemplo de script R criando um modelo dentro do módulo Execute R Script

## Visualizando todas as bibliotecas do R instaladas no AzureML

Antes de baixar bibliotecas do CRAN e instalá-las (conforme demonstrado anteriormente neste capítulo), vale a pena verificar se as mesmas já não estão disponíveis na plataforma.

Para tal basta executar o seguinte script em R (dentro do Execute R Script):

```
result <- data.frame(installed.packages())
maml.mapOutputPort("result")
```

## O módulo Create R Model

No cenário anterior já discutimos que é possível criar um experimento completo de aprendizagem de máquina dentro do módulo Execute R Script. Contudo, o modelo neste cenário existe apenas dentro do script e a interação com os outros componentes do AzureML é reduzida.

O módulo *Create R Model* cria um modelo não treinado originado de um script em R que é fornecido na sua criação. Após isso é possível usar a saída deste módulo (modelo não treinado / *untrained model*) como a saída de qualquer outro algoritmo (*learner*) que está incluso no ambiente do AzureML. Veja a comparação da saída do módulo Create R Model com o de SVM (Two-Class Support Vector Machine) na Figura 201, perceba que é exatamente o mesmo tipo.

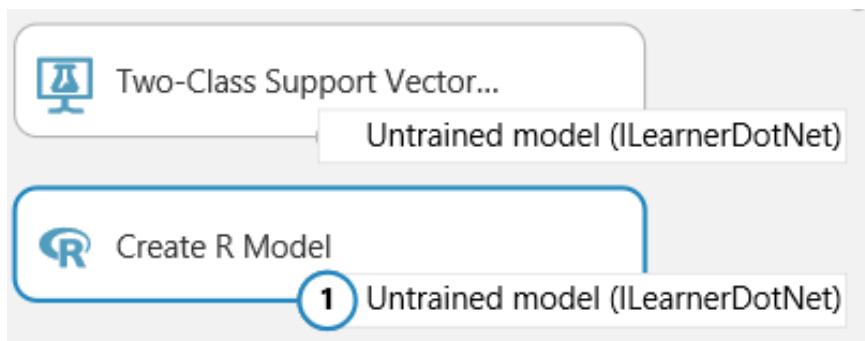


Figura 201 – Saídas do trainer e do Create R Model

Este módulo também faz integração com o módulo Score Model, ou seja, quando este modelo (*Create R Model*) for aplicado a este módulo (*Score Model*), o que é executando também é definido no módulo. A Figura 202 a seguir retrata os parâmetros necessários a este módulo.

#### ▲ Create R Model

Trainer R script

```
1 # Input: dataset
2 # Output: model
3
4 # The code below is an example which can be replaced with
5 # See the help page of "Create R Model" module for the
6
```

Scorer R script

```
1 # Input: model, dataset
2 # Output: scores
3
4 # The code below is an example which can be replaced with
5 # See the help page of "Create R Model" module for the
6
```

Figura 202 - Parâmetros necessários ao Create R Model

Como visto no Capítulo 4, basta utilizar um módulo *Train Model* para treinar este modelo gerado no R sobre um *dataset* do AzureML, e depois utilizar o módulo *Score Model* para fazer suas previsões (usando o código customizado) e seguir o mesmo fluxo de outros experimentos.

Importante observar que o modelo é posto em cache após a sua primeira execução e o módulo não é invocado nas execuções seguintes

No momento em que este livro foi escrito ainda não há possibilidade de integrar o *Create R Model* com os módulos *Evaluate Model* ou *Cross-Validate Model*, apenas *Train Model* e *Score Model* são suportados nesta versão.

## Configurando o módulo Create R Model para criar um modelo de Naïve Bayes

Vimos na sessão anterior que necessitamos informar dois parâmetros: *Trainer R Script* e o *Scorer R Script* para a utilização do módulo de modelo customizado em R.

Abaixo temos um exemplo utilizando a biblioteca *e1071* que contém o algoritmo de *Naïve Bayes* implementado. Este é o exemplo padrão do componente e com ele é possível observar o script necessário.

*Trainer R Script*

# Carrega a biblioteca que contém o algoritmo Naïve Bayes

*library(e1071)*

# A função pré-definida, *get.label.columns()*, retorna a coluna que foi marcada como sendo a classe no módulo de Train Model.

*features <- get.feature.columns(dataset)*

# A função pré-definida, `get.feature.columns()`, retorna as colunas que foram marcadas como características nos metadados do dataset. Por padrão, todas as colunas que não sejam a coluna que foi marcada como sendo a de classe são consideradas colunas de características.

```
labels <- as.factor(get.label.column(dataset))
```

# Cria um dataframe com as características e classes

```
train.data <- data.frame(features, labels)
```

# Nas linhas abaixo a função pré-definida, `get.feature.column.names()`, é usada para recuperar os nomes das colunas de características do dataset. Também é atribuído um nome temporário ("Class") para a coluna de classes.

```
feature.names <- get.feature.column.names(dataset)
```

```
names(train.data) <- c(feature.names, "Class")
```

#executa de fato o algoritmo de Naïve Bayes, sobre o dataframe criado no início do script, e marca todas as colunas dele como entrada para a coluna de classes. O ponto ":" indica todas as colunas, e o nome de coluna antes do til "~" indica a coluna com as classes.

```
model <- naiveBayes(Class ~ ., train.data)
```

O modelo precisa, obrigatoriamente, ser armazenado em uma variável de nome "model".

---

Scorer R Script

```
# Carrega a biblioteca em memória  
library(e1071)  
  
# Calcula as probabilidades predizidas para o dataset de scoring usando o modelo  
treinado no script de treino. O modelo treinado foi armazenado na variável "model", e  
ela é acessível neste script.  
probabilities <- predict(model, dataset, type="raw")[,2]  
  
# Aplica um limiar (threshold) de 0.5 para atribuir a classe prevista. Isso pode e deve ser  
ajustado pelo cientista de dados a modo de tornar o script mais adequado ao problema  
em questão.  
classes <- as.factor(as.numeric(probabilities) >= 0.5)  
  
# Combina as classes com o resultado previsto e armazena na variável "scores"  
scores <- data.frame(classes, probabilities)
```

É obrigatório que o resultado do *Scorer R Script* seja armazenado em uma variável chamada “scores”. Com isso será utilizado este script durante o processo de *scoring* do seu modelo.

Neste capítulo buscamos aproximar vocês dos conceitos básicos da linguagem R. Não buscamos o aprofundamento completo da linguagem, mas sim uma introdução daquilo que consideramos o básico e também objetivando facilitar o entendimento de todos os códigos em R que foram apresentados no livro.

Assim como qualquer linguagem de programação, a melhor forma de aprender o R é através da prática. Durante o seu dia a dia você será confrontado com cenários aonde as soluções podem não ser triviais. Contudo, entendendo a base, conforme demonstramos neste capítulo, acreditamos que é possível lidar com todos estes casos.

## 10 – Visualização de Dados

Uma das áreas importantes dentro das atividades que os cientistas de dados fazem é analisar visualmente os dados, para facilitar a exploração e insights que possam surgir, e que analisando os dados em formato tabular nem sempre são tão simples de se enxergar. Levando este ponto em consideração, dentro do *Azure Machine Learning* é possível visualizar como os dados estão visualmente dispostos e como podem interagir com os meios externos. Neste capítulo serão apresentadas duas formas de interagir com os dados, uma diretamente dentro do *Azure Machine Learning* e outra com o seu consumo dentro do Microsoft Power BI<sup>64</sup>.

A primeira forma proposta de visualizar os dados é dentro do próprio *Azure Machine Learning*, onde é possível visualizar uma saída da execução de determinados módulos, como o *Evaluate Model* ou um *Execute R Script*. Com base nestas possibilidades, a primeira forma que será explorada é a saída visual de comparação de duas classificações binárias dentro de um mesmo experimento. Para maiores detalhes de como funcionam estes resultados, recomendo que leia o Capítulo 5 sobre validação de modelos.

Para chegar até esta forma de visualização dos dados, você precisará abrir um experimento já criado na galeria do *Azure Machine Learning*. Procure por "*Binary Classification: Credit risk prediction*", que é um experimento de classificação binária que analisa risco de empréstimo financeiro com base em um *dataset* fornecido por um banco alemão nos anos 2000<sup>65</sup>. Veja a Figura 203 como é a seleção do experimento comentado.

---

64 [www.powerbi.com](http://www.powerbi.com)

65

[https://archive.ics.uci.edu/ml/datasets/Statlog+\(German+Credit+Data\)](https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data))

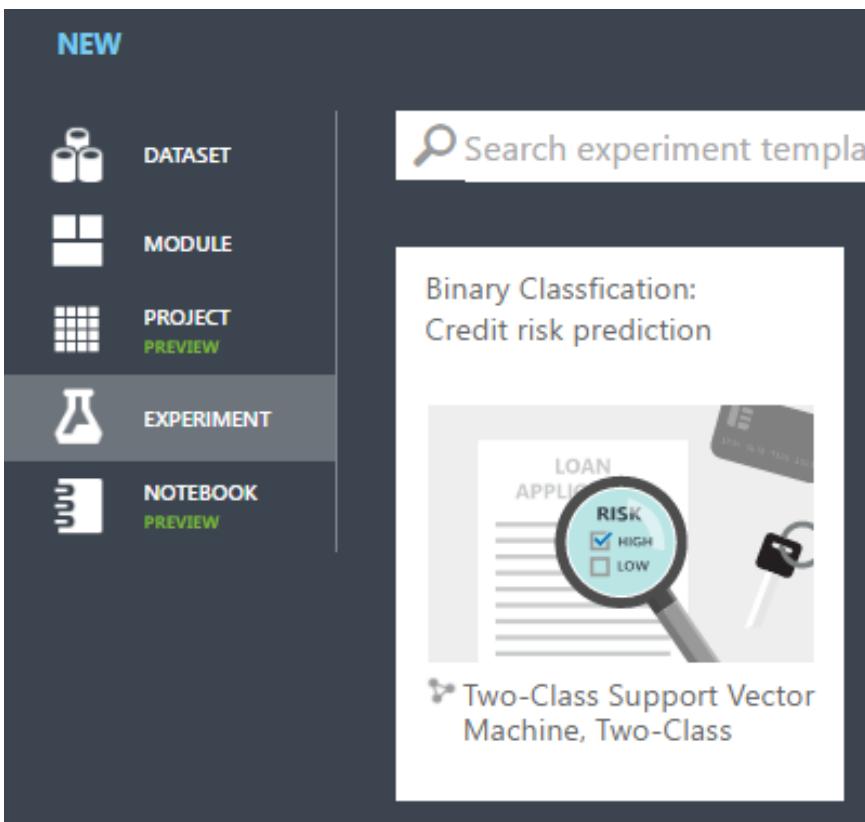


Figura 203 - Escolhendo o Experimento

Após escolher e abrir o experimento, percorra a página com todas as ligações entre as tarefas até onde está o módulo *Evaluate Model*. Acompanhe na Figura 204 o experimento completo aberto e na Figura 205 o módulo de avaliação.

## Binary Classification: Credit risk prediction

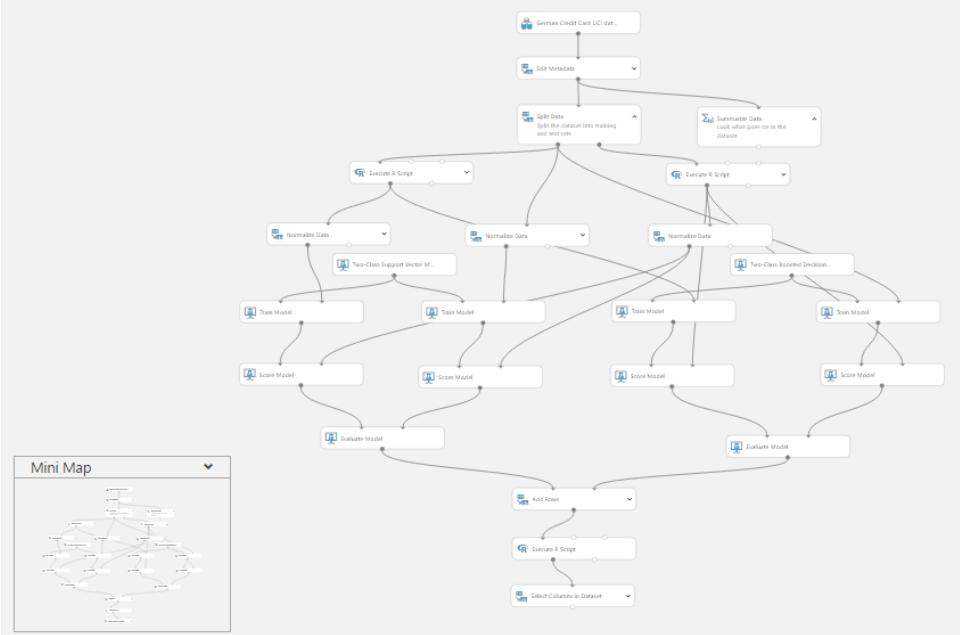


Figura 204 - Experimento completo

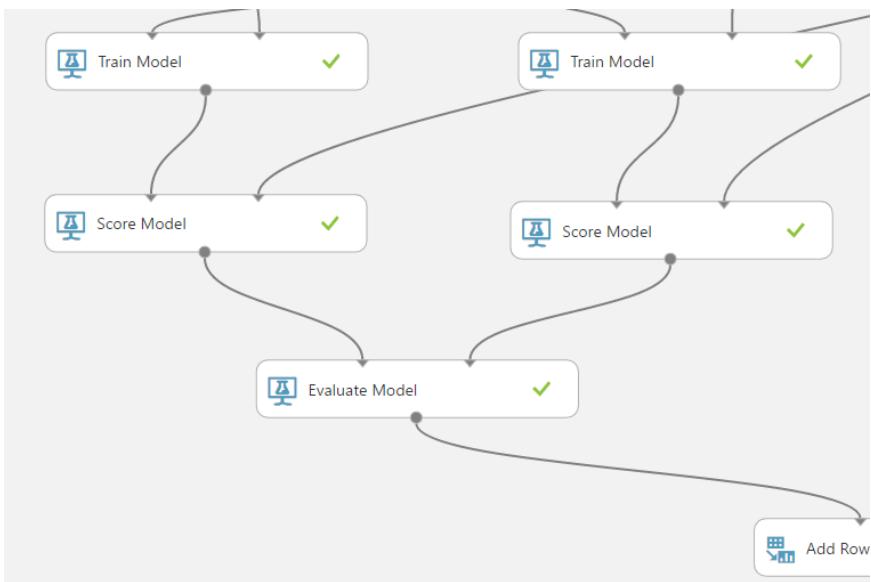


Figura 205 - Módulo de Evaluate Model

Repare que ao clicar na saída do módulo *Evaluate Model* a opção *Visualize* está disponível. É nesta opção que você deve clicar, para ver o resultado da avaliação de ambos modelos que foram utilizados para esta tarefa do experimento. Caso seu modelo não tenha a parte de *Visualize* habilitada, conforme a Figura 206, você deve rodar o modelo e então verificar esta opção de saída.

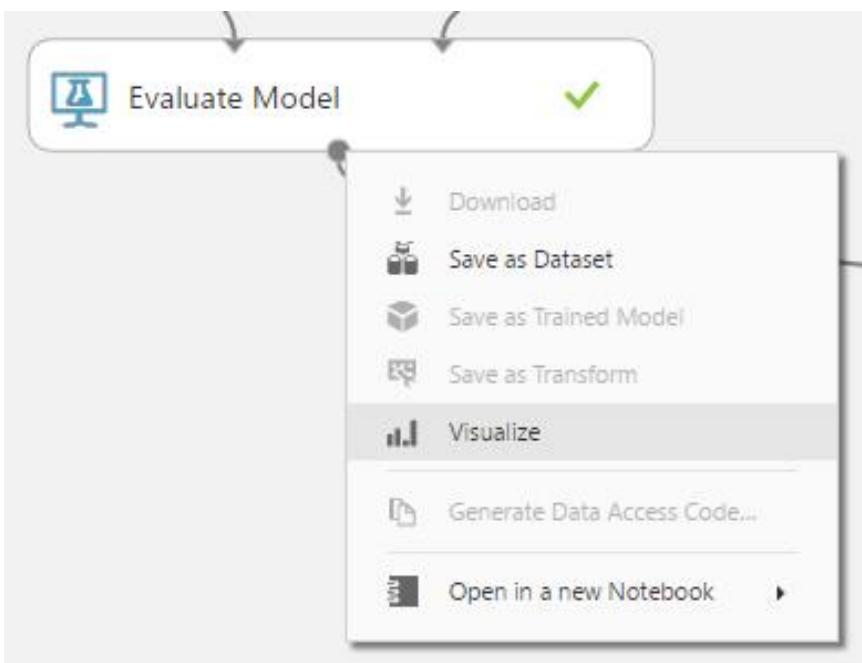


Figura 206 - Opção de Visualização da tarefa Evaluate Model

O resultado esperado é que, a princípio, seja apresentada a curva ROC de comparação entre ambos algoritmos. Lembrando que o Capítulo 5 aborda esta explicação. Esta saída visual possibilita que nós, cientistas de Dados, possamos encontrar qual resultado é melhor para nosso experimento de forma rápida e precisa.

Uma segunda forma de analisar dados dentro das tarefas do *Azure Machine Learning*, pode ser com as implementações específicas em linguagem R ou então Python. Seguindo esta linha de pensamento, vamos criar um experimento novo que receberá um conjunto de dados de exemplo e com um script R os dados serão trabalhados para segmentar os dados em grupos com o algoritmo de Cluster K-Means. Este algoritmo é nativo na Linguagem R. Para começar, inicie um experimento em branco, como a Figura 207.

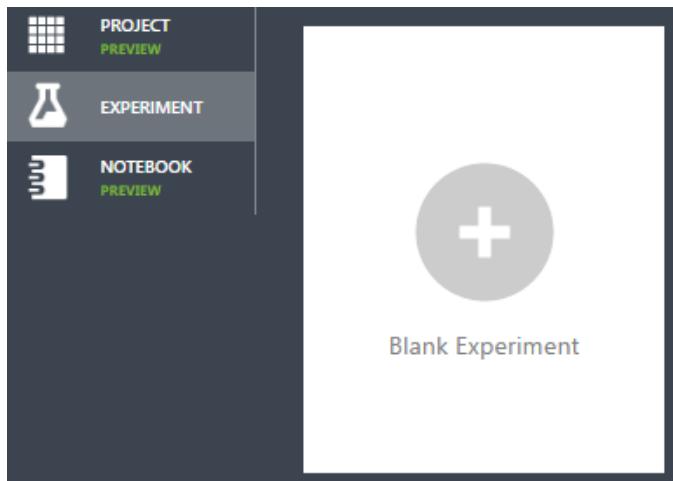


Figura 207 - Criando um experimento em branco

A primeira tarefa que será criada neste experimento é a leitura de dados. Que neste caso, vamos voltar ao Capítulo 9 onde explicamos um pouco da linguagem R com a explicação dos códigos que foram apresentados anteriormente no Capítulo 1 sobre o dia a dia do Cientista de Dados. Nestas partes do livro, foi apresentada uma implementação sobre os clusters com *K-Means*. É hora de voltar até aqueles códigos que geram os dados de entrada para a análise de Clusters de Investimento Vs. Faturamento. Para exemplificar a criação customizada de componentes visuais dentro do *Azure Machine Learning*, os dados que foram gerados aleatoriamente no exemplo citado foram salvo em um arquivo CSV e colocado dentro de um *Azure Blob*<sup>66</sup> para que seja consumido pelo módulo *Import Data*. Acompanhe na Figura 208 este processo.

---

<sup>66</sup>Azure Blob: <https://azure.microsoft.com/en-us/services/storage/blobs/>

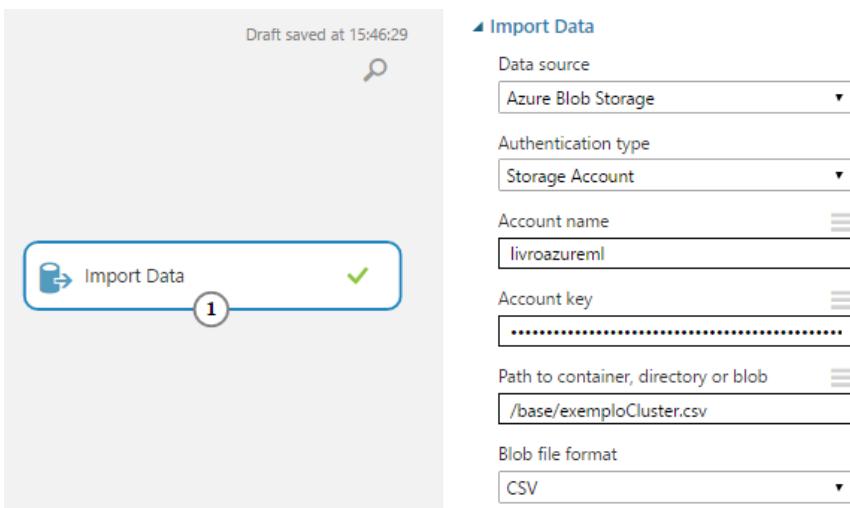


Figura 208 - Importando dados para o experimento

A configuração desta tarefa pode ser encontrada no Capítulo 3 sobre o trabalho com dados externos. Após configurar o componente, e visualizar se a importação funcionou corretamente, as 200 linhas que foram adicionadas no arquivo à partir do código referenciado podem ser visualizados, conforme a Figura 209.

[Interação com R e PowerBI](#) › [Import Data](#) › [Results dataset](#)

rows	columns
200	2
<hr/>	
	investimento faturamento
<hr/>	
view as	 
	
5.66	11.8
5.21	10.47
4.97	12.93
7.17	11.06
5.38	10.03
3.46	7.03

Figura 209 - Resultado da importação dos dados

Se a importação dos seus dados ocorreu conforme planejado, você terá um resultado similar ao apresentado na Figura 209. O passo seguinte é adicionar o módulo *Execute R Script* no seu experimento e fazer a ligação entre a saída do módulo *Import Data* e a primeira entrada do módulo *Execute R Script*. A Figura 210 apresenta como esta ligação deverá ficar.

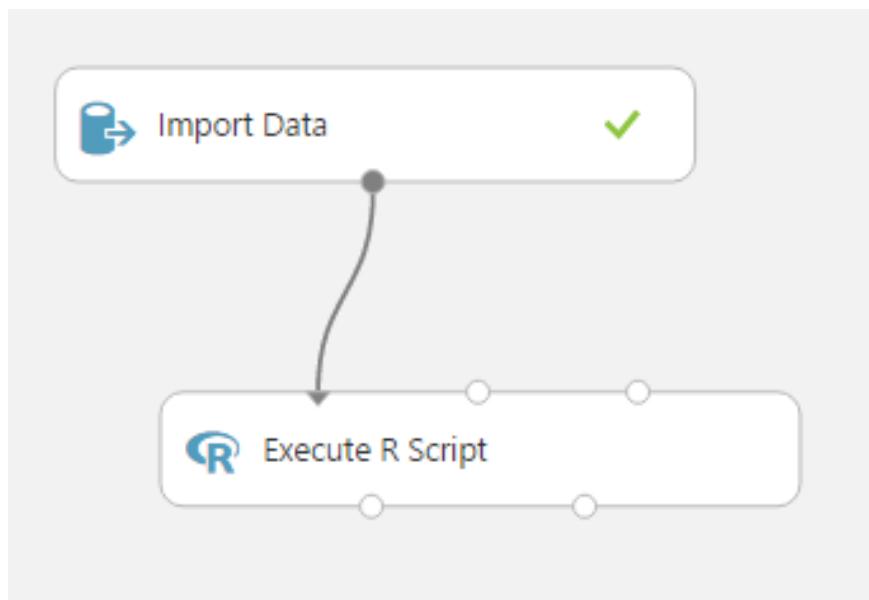


Figura 210 - Conectando os dados importados ao script em R

Se clicar no módulo *Execute R Script*, as configurações do componente serão abertas no menu da direita, permitindo que seja editado o script que deseja trabalhar. Remetendo novamente ao código gerado anteriormente, podemos reaproveitar a parte de criação do cluster e plotagem do gráfico. Neste caso, para facilitar a escrita do código, acompanhe o bloco a seguir:

```
dataset1 <- maml.mapInputPort(1)

#install.packages("RColorBrewer")

library(RColorBrewer)

cores <- brewer.pal(10, "Paired")

dados2 <- dataset1[c("investimento", "faturamento")]
```

```

grupos2 <- kmeans(dados2,3)

lot(dados2, col=cores[grupos2$cluster], pch=19
    , main="Investimento X Faturamento"
    , xlab="Investimento em Marketing (vezes 10)"
    , ylab="Faturamento em R$ (vezes 100)")

dados2$cluster <- grupos2$cluster

maml.mapOutputPort("dados2");

```

Com este código é possível receber os dados originais do módulo *Import Data*, dividir os dados em 3 grupos (a quantidade de 3 grupos foi escolhida apenas para elucidar o exemplo), gerar o gráfico baseado nestes dados e então escrever para a saída de dados os valores que foram atribuídos pelo algoritmo *K-Means*. O código dentro do componente pode ser encontrado na Figura 211.

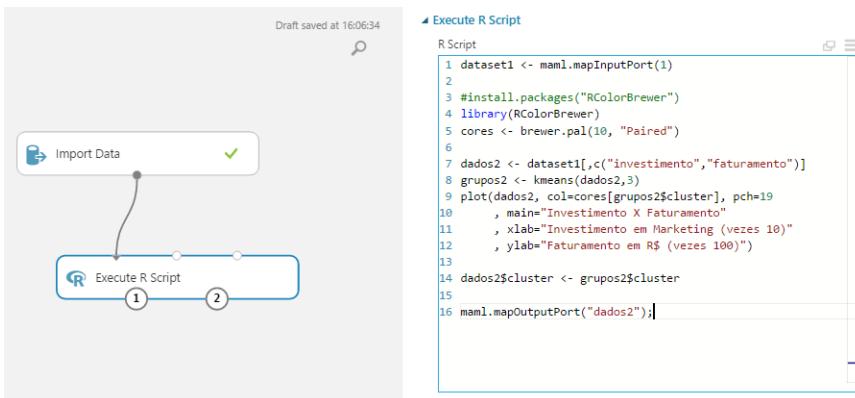


Figura 211 - Código em R no componente

Ao executar o experimento completo, as duas saídas do módulo *Execute R Script* terão resultados. A primeira saída, representada pelo número 1 na Figura 211, terá a saída em formato tabular dos dados. Esta saída é preenchida pelo código que está na linha 16, também na Figura 211. Acompanhe nas Figuras 212 e 211 o resultado.

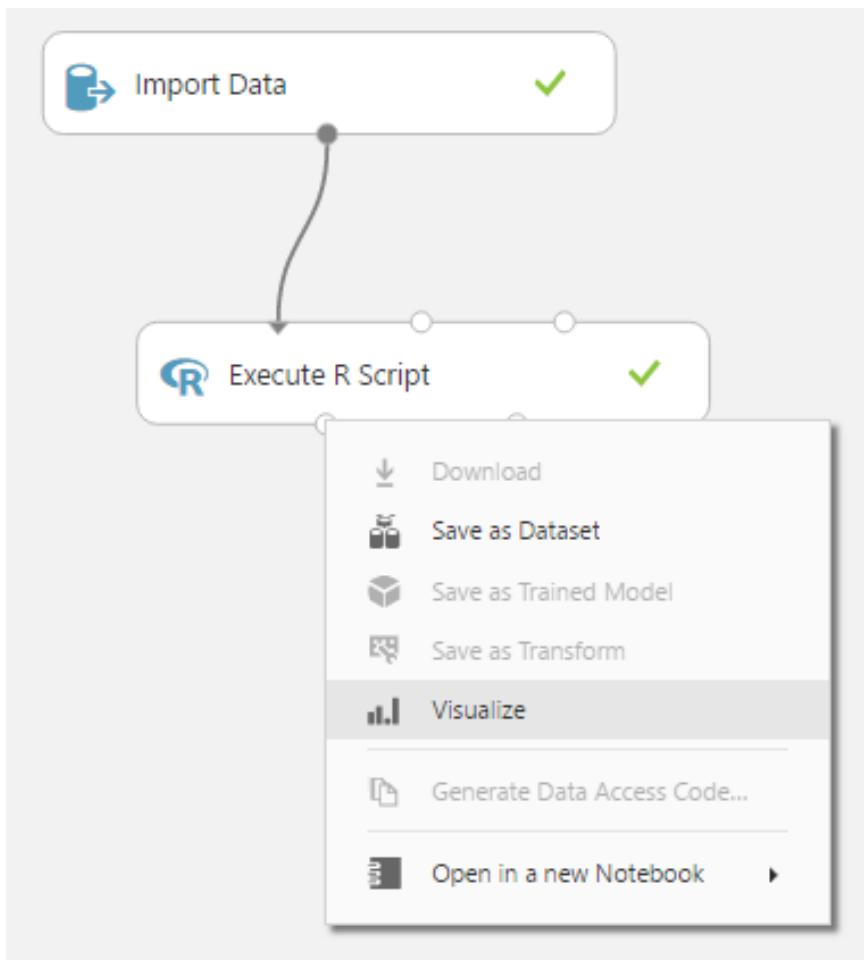


Figura 212 - Visualização da primeira saída

Interação com R e PowerBI ➤ Execute R Script ➤ Result Dataset

rows	columns		
200	3		
<hr/>			
	investimento	faturamento	cluster
view as			
	5.66	11.8	2
	5.21	10.47	3
	4.97	12.93	2
	7.17	11.06	2
	5.38	10.03	3

Figura 213 - Resultado da primeira saída

Já a segunda saída, representada pelo número 2 (Figura 211), apresenta a parte processada pelo script em linguagem R que foi colocado no componente. Neste caso, como o código possui uma instrução de plotagem, esta pode ser encontrada nesta saída. Veja nas Figuras 214 e 215 como ficaram estas saídas.

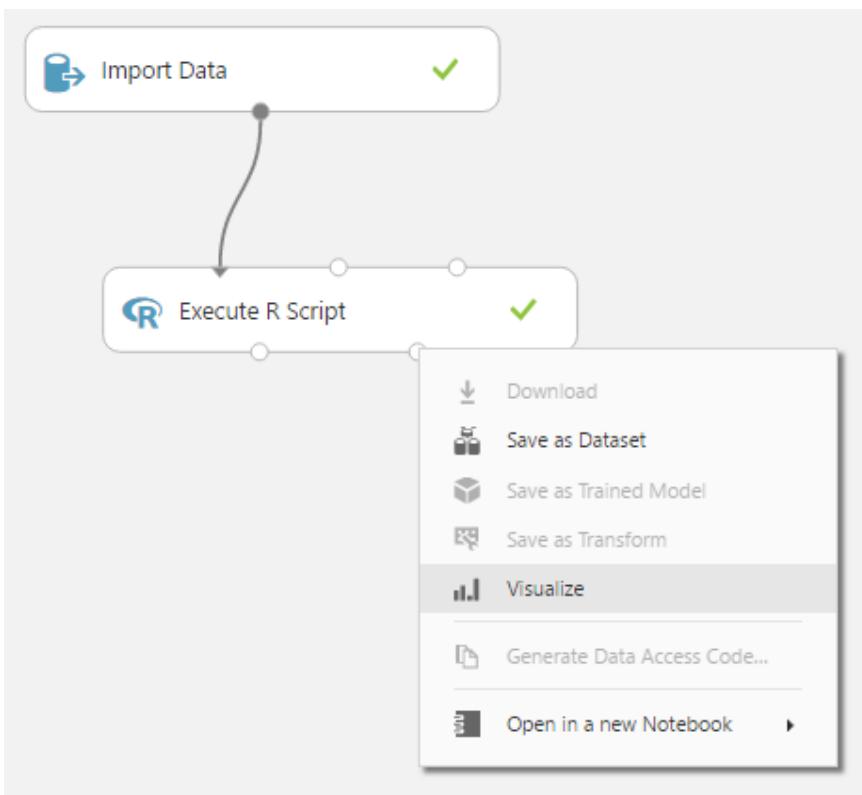


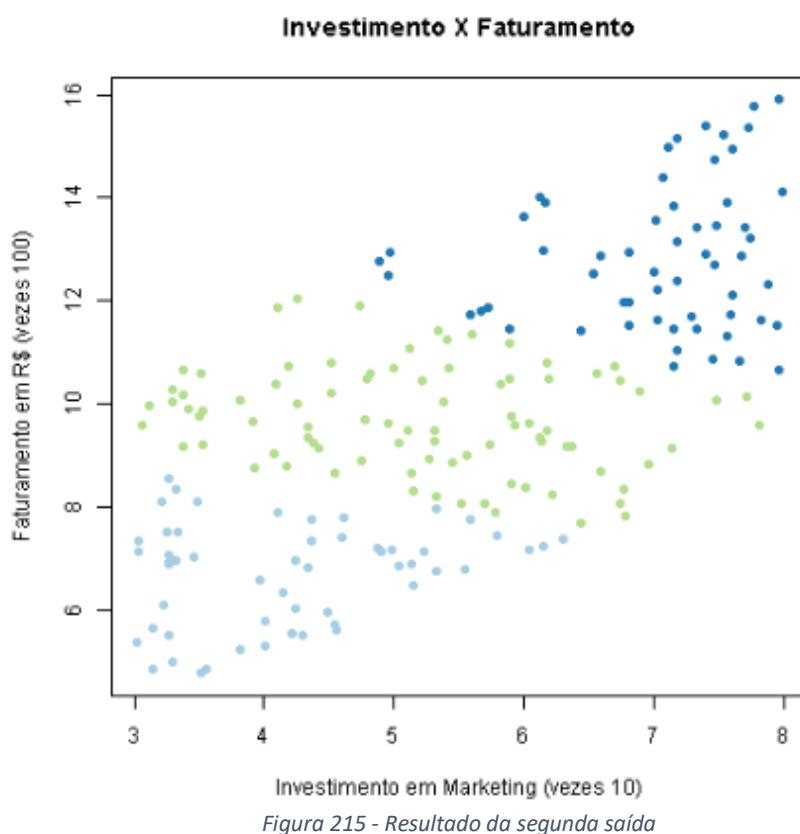
Figura 214 - Visualização da segunda saída

Interação com R e PowerBI ➤ Execute R Script ➤ R Device

▲ Standard Error

R reported no errors.

▲ Graphics



Encontrar esta apresentação visual na saída do módulo *Execute R Script* não é, e não deve ser, o fim de sua visualização de dados. Uma extensão

que funciona bem para explorar visualmente os dados que foram gerados pelos experimentos, é o consumo dos dados pelo Microsoft Power BI. Não foi por acaso que adicionamos uma nova coluna na saída dos dados do código em R que colocamos no componente. Volte até a Figura 211 e observe a linha indicada com o número 14. Ela adiciona uma coluna chamada Cluster no dataset e você conseguirá ver os dados desta coluna na Figura 213.

Já que o componente entrega os dados que precisamos para trabalhar, um componente de escrita externa deve ser colocado, para então, enviar esses dados de saída para outra destinação. Neste experimento o módulo *Export Data* foi adicionado e configurado para escrever os dados no *Azure SQL Database*<sup>67</sup>. Veja o componente novo na Figura 216.

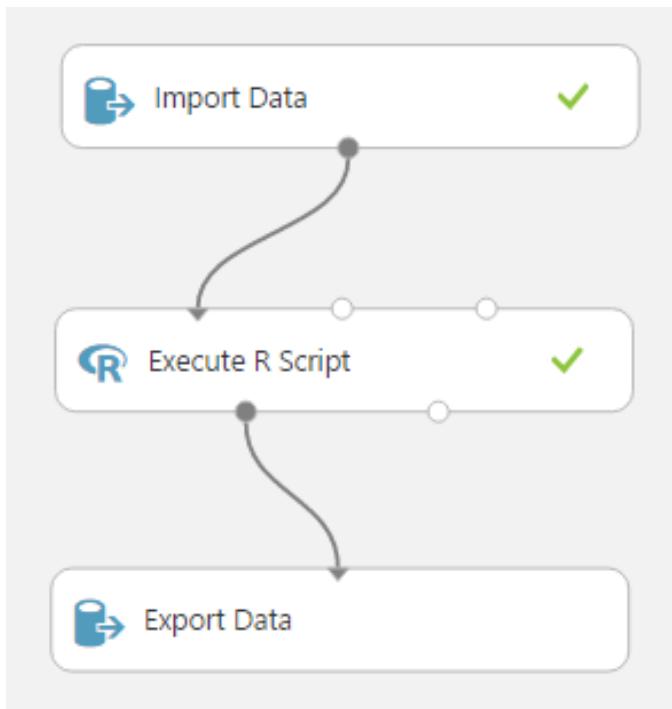


Figura 216 - Escrevendo o resultado do experimento

67 <https://azure.microsoft.com/en-us/services/sql-database/>

Os dados exportados já foram explicados no Capítulo XX, sobre Trabalhar com Dados Externos. Caso tenha alguma dúvida, vale a pena revisar este processo. Na Figura 217 é apresentada a forma como foi configurado o módulo no experimento que criamos.

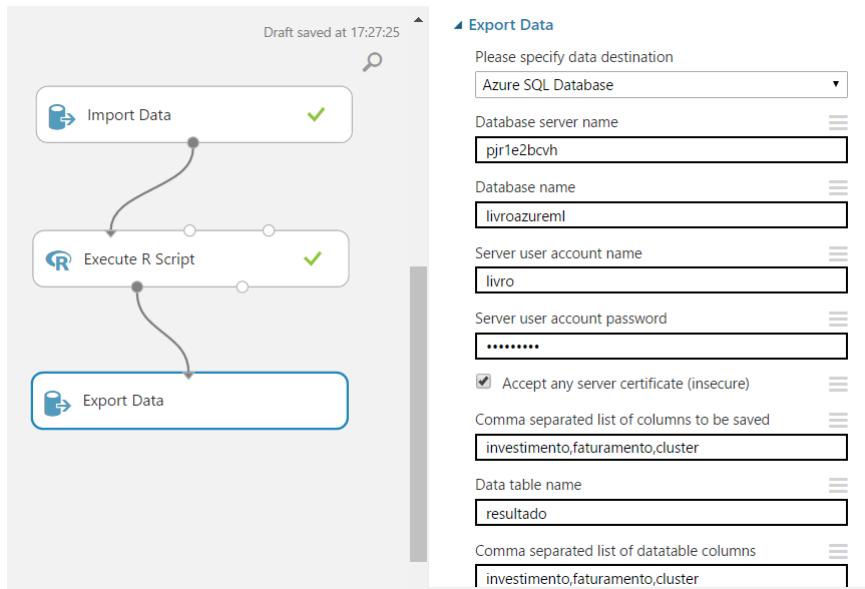


Figura 217 - Configuração da exportação dos dados

A tabela que foi criada, para receber os dados, segue a estrutura como apresentada no bloco de código T-SQL a seguir:

```
CREATE TABLE [dbo].[resultado](
    [investimento] [decimal](5, 2) NULL,
    [faturamento] [decimal](5, 2) NULL,
    [cluster] [int] NULL
```

)

Se o experimento executar corretamente, os dados serão escritos na tabela do *Azure SQL Database*, e você terá um resultado conforme o apresentado na Figura 218 de dentro do *SQL Server Management Studio*.

The screenshot shows a window titled "SQLQuery2.sql - pj...azureml (livro (97))". The query displayed is:

```
SELECT TOP 200 [investimento]
      ,[faturamento]
      ,[cluster]
  FROM [dbo].[resultado]
```

The results pane shows a table with 9 rows of data:

	investimento	faturamento	cluster
1	5.66	11.80	2
2	5.21	10.47	1
3	4.97	12.93	2
4	7.17	11.06	2
5	5.38	10.03	1
6	3.46	7.03	3
7	7.81	9.59	1
8	3.14	4.85	3
9	6.76	8.34	1

Figura 218 - Resultado do experimento dentro do Azure SQL Database

Para aproveitar estes dados gerados pelo *Azure Machine Learning* e salvos no banco de dados, uma forma robusta é utilizar o Power BI. Ao abrir o

Power BI Desktop<sup>68</sup>, clique no ícone Obter Dados, conforme apresentado na Figura 219.

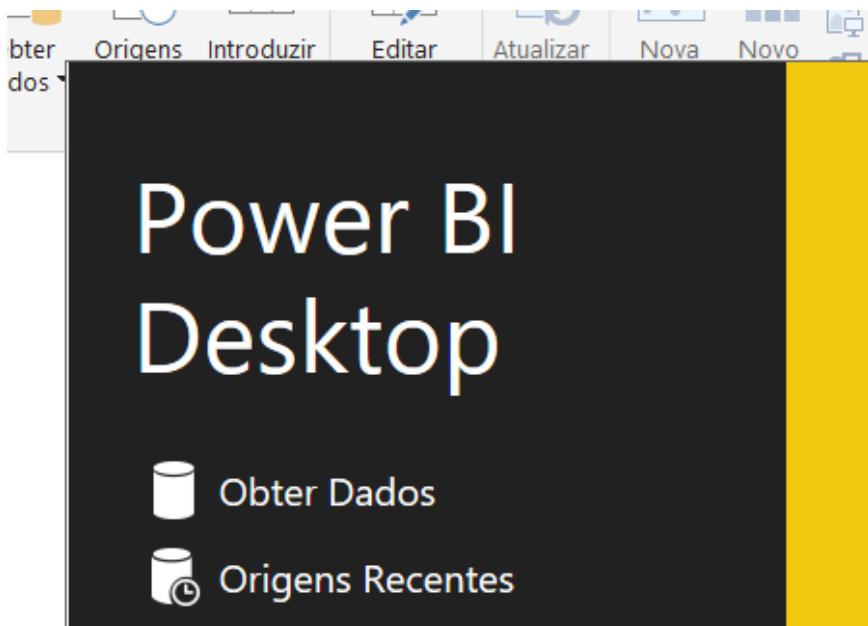


Figura 219 - Obtenção de dados do Power BI

Na janela que se abrirá, aponte nas opções da esquerda a seleção de *Azure* e em seguida, nas opções da direita, aponte para *Bases de Dados do Microsoft Azure*. Acompanhe a seleção na Figura 220.

---

<sup>68</sup> Link de Download do PowerBI Desktop:  
<https://powerbi.microsoft.com/pt-br/desktop/>

## Obter Dados

Procurar

Tudo

Ficheiro

Base de Dados

Azure

Serviços Online

Outras

Azure

- Base de Dados SQL do Microsoft Azure
- Microsoft Azure SQL Data Warehouse
- Microsoft Azure Data Marketplace
- Microsoft Azure HDInsight
- Armazenamento de Blobs do Microsoft Azure
- Armazenamento de Tabelas do Microsoft Azure
- Azure HDInsight Spark (Beta)
- Microsoft Azure DocumentDB (Beta)
- Arquivo Data Lake do Microsoft Azure (Beta)

Figura 220 - Selecionando o Azure SQL Database

Ao clicar em ligar no final da tela, novas opções de interação são apresentadas, solicitando alguns dados de conexões. Neste momento, é importante deixar marcada a opção de *DirectQuery*. Veja como é esta tela, na Figura 221.

# Base de Dados do SQL Server

Importe dados de uma base de dados do SQL Server.

Servidor

pjr1e2bcvh.database.windows.net,1433

Base de Dados (opcional)

livroazureml

Importar

DirectQuery

> Opções avançadas

Figura 221 - Conectado ao servidor do Azure SQL Database

Ao avançar para a tela seguinte, as credenciais de autenticação são solicitadas. Neste caso, é necessário informar o usuário e senha para acesso à base de dados. Repare na alteração da forma de autenticação no lado esquerdo da tela, na Figura 222.

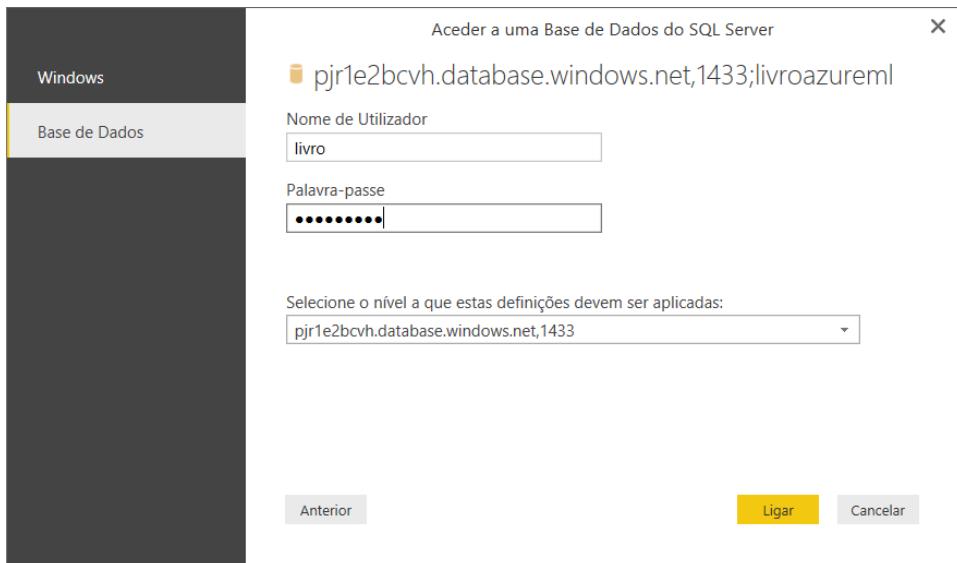


Figura 222 - Autenticação com o SQL na base de dados

Em seguida é apresentada a lista de tabelas existentes no banco de dados que foi conectado. Ao clicar em uma ou mais tabelas, a lista de registros encontrados em cada entidade é apresentada. Acompanhe esta visualização na Figura 223.

### Navegador

Opções de Apresentação ▾

- pj1e2bcvh.database.windows.net,1433: livroaz...
- sys.database\_firewall\_rules
- resultado**

investimento	faturamento	cluster
5.66	11.8	2
5.21	10.47	1
4.97	12.93	2
7.17	11.06	2
5.38	10.03	1
3.46	7.03	3
7.81	9.59	1
3.14	4.85	3
6.76	8.34	1

Figura 223 - Dados existente na tabela

Ao final, basta clicar no botão Carregar no final da tela, e aguardar os atributos da entidade serem apresentados na região de campos do PowerBI. Confira a Figura 224 onde estes campos estão disponíveis.

The screenshot shows the 'Campos' (Fields) pane of the Power BI interface. On the left, under 'Visualizações' (Visualizations), there is a grid of icons representing various chart types. Below this are two buttons: a dashed grid icon labeled 'Valores' (Values) and a paint roller icon labeled 'Filtros' (Filters). In the center, a search bar has 'Procurar' (Search) written in it. To the right of the search bar is a list of fields grouped under 'resultado': 'cluster', 'faturamento', and 'investimento'. At the bottom of the pane, there are three dashed boxes labeled 'Arrastar os campos de dad...' (Drag fields from data...) corresponding to the 'Valores', 'Filtros', and 'Relatório' sections.

Figura 224 - Campos disponíveis para se trabalhar

Agora que os dados já estão prontos para serem consumidos dentro do Power BI, é hora de começar a montar o relatório. Neste primeiro momento, serão criados três *cards* para apresentar dados sumarizados dos campos que retornaram da nossa base. Para isso, clique no campo *cluster* dentro dos campos disponíveis. Ele ficará com o *checkbox* marcado e aparecerá um gráfico na área de edição do relatório. Veja como ficou na Figura 225.

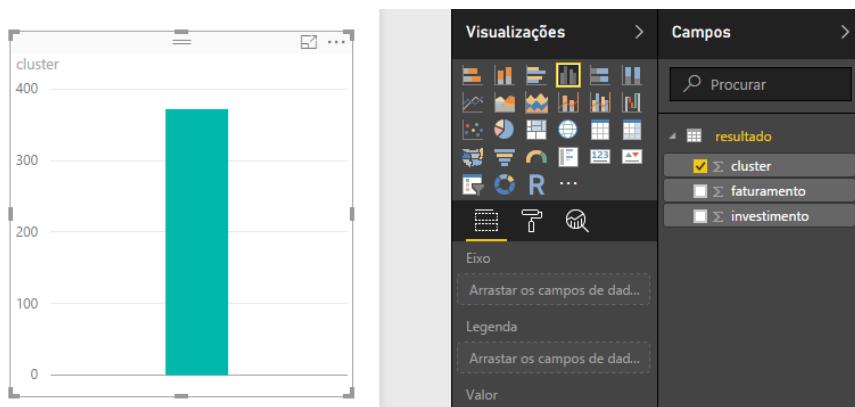


Figura 225 - Inserção do primeiro valor no PowerBI

Reparam que o dado é apresentado em formato de gráfico, porque na área central superior da Figura 225, o elemento de visualização que está selecionado é o Gráfico de Colunas Agrupadas. Isso acontece porque o Power BI inferiu que esta forma visual seria a mais apropriada para apresentar este dado. Para alterar e deixar da forma como queremos, mude a Visualização deste gráfico para Cartão. O Cartão é representado por um ícone que tem os números 123 e está localizado na quinta coluna da penúltima fileira de ícones. Acompanhe este elemento visual na Figura 226.

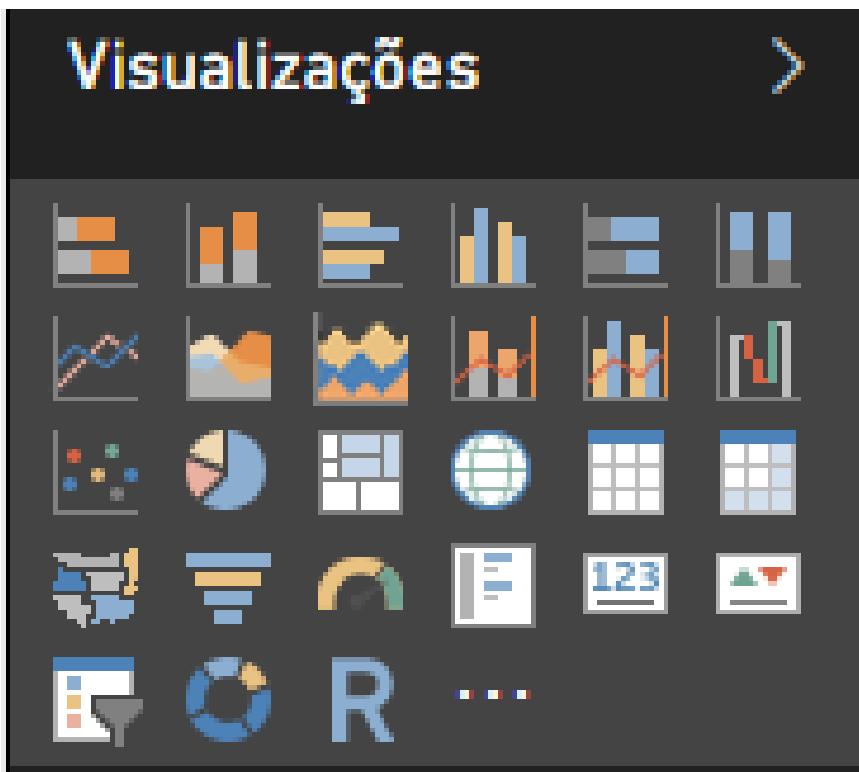


Figura 226 - Tipos de Visualizações

Ao manter o gráfico selecionado na área de edição do relatório e alterar o elemento visual para o Cartão, você terá uma alteração visual no que é apresentado como resultado. Veja como ficou o cartão com os dados do cluster na Figura 227.

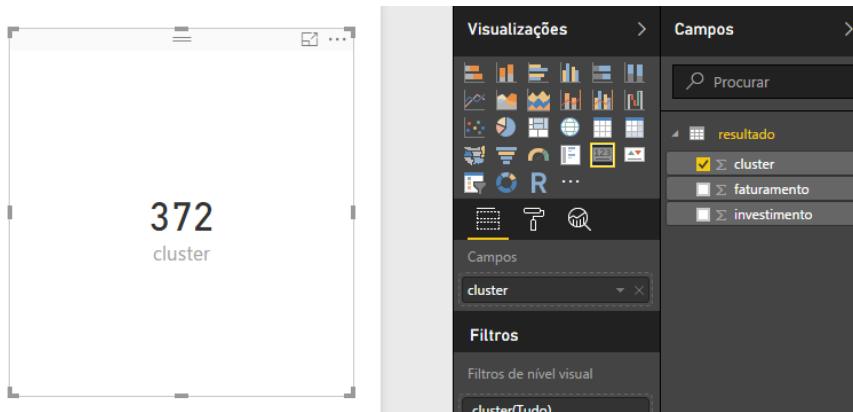


Figura 227 - Mudando a visualização para Cartão

Neste momento, a informação que esperamos receber de volta, é a quantidade de clusters que existem na base de dados. E como nós sabemos que são apenas 3 clusters, então algo está errado nesta apresentação. Para corrigir este erro, mantenha selecionado o cartão que acabou de alterar e vá até as opções de agregação do campo abaixo dos elementos de Visualizações. Clique na seta para baixo do cluster e modifique a forma de agregação. Mude para a opção Contagem (Distinta). Ao alterar, o valor do Cartão de clusters já estará correto. Veja como é na Figura 228.

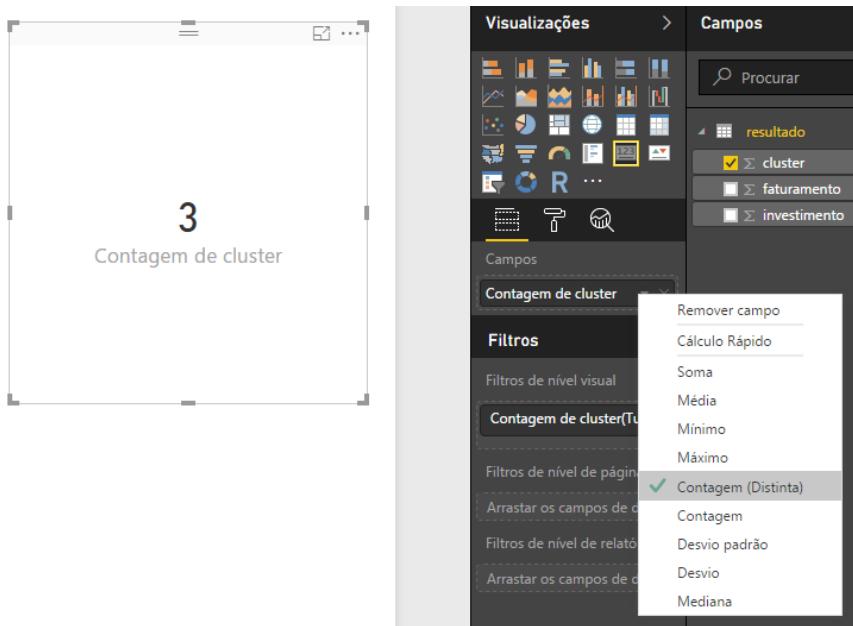


Figura 228 - Mudança do fator de agregação do cartão

Neste momento, o valor da quantidade de clusters está correto. Altere o tamanho do cartão, aumentando ou diminuindo através das áreas de seleção expostas nas bordas do mesmo. Deixe ele menor, e clique em qualquer área em branco da região de edição do relatório. Assim, você poderá adicionar novos elementos visuais. Faça este procedimento e adicione mais dois cartões. Um para a somatória de Faturamento e outro para a Média de Investimento. Veja a Figura 229 como ficará este relatório.

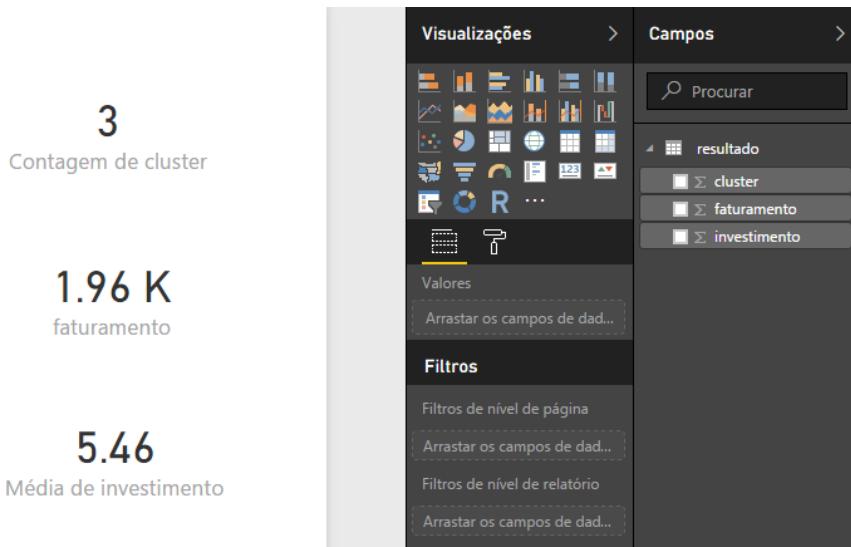


Figura 229 - Três cartões com informações sobre os dados

Um elemento interessante para poder explorar os dados, é o segmentador. Com este elemento é possível filtrar os dados no nível de apresentação. Este elemento visual está posicionado na primeira coluna da última linha de ícones. Para adicionar este elemento ao relatório, mais uma vez selecione alguma área em branco da região de edição de relatórios e clique no campo cluster. Altere a forma de visualização deste elemento e posicione próximo aos elementos visuais de cartões que já criou. Veja a Figura 230 como ficou o resultado.

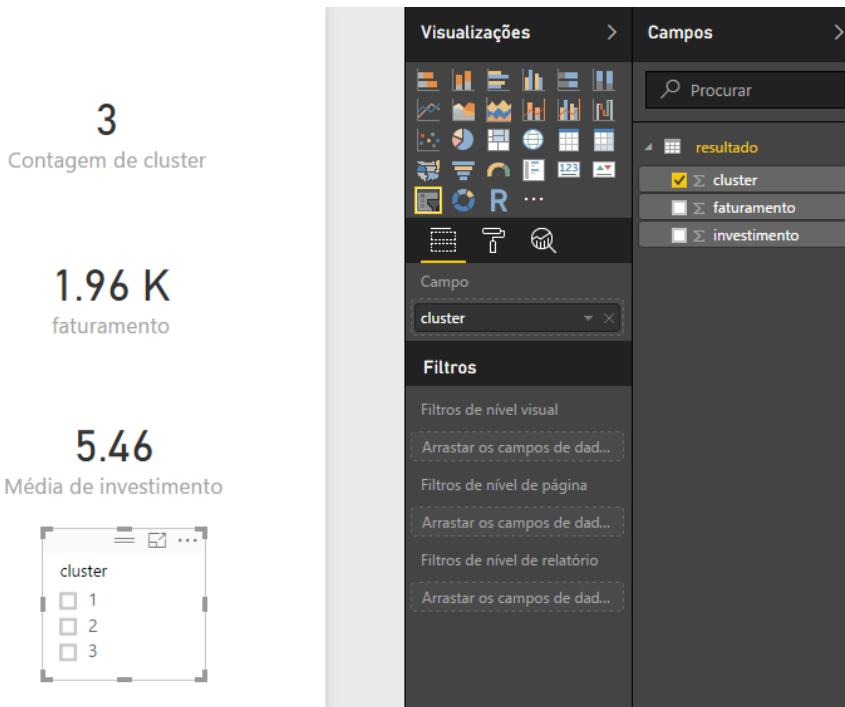


Figura 230 - Adicionando segmentador ao relatório

Clique nos *checkboxes* do segmentador e veja os dados serem filtrados com base no que selecionou.

Por fim, vamos adicionar um elemento visual da Linguagem R no relatório. Para isso, clique em uma área em branco do relatório e selecione os três campos existentes na área de campos do relatório. Garanta que o gráfico de colunas agrupadas que foi montado esteja selecionado e então altere o componente visual para o *R Script Visual*. Este elemento é o terceiro ícone existente na última linha. Ele é representado pela letra R. Ao clicar neste componente, um alerta pode ser apresentado para seu relatório. Confirme que quer escrever os códigos para apresentar os dados através do R.

Em seguida o gráfico irá desaparecer e uma área de edição de códigos será aberta na parte inferior do Power BI Desktop. Esta área recebe o código

em R que irá gerar os elementos visuais com base nos dados que recebeu da origem. Escreva o código que está no bloco a seguir.

```
library(RColorBrewer)  
  
cores <- brewer.pal(10, "Paired")  
  
plot(dataset[,c("investimento", "faturamento")], col=cores[dataset$cluster], pch=19)
```

Ao adicionar este código R no componente, e reorganizar o relatório colocando também um título, é possível obter algo como o demonstrado na Figura 231.

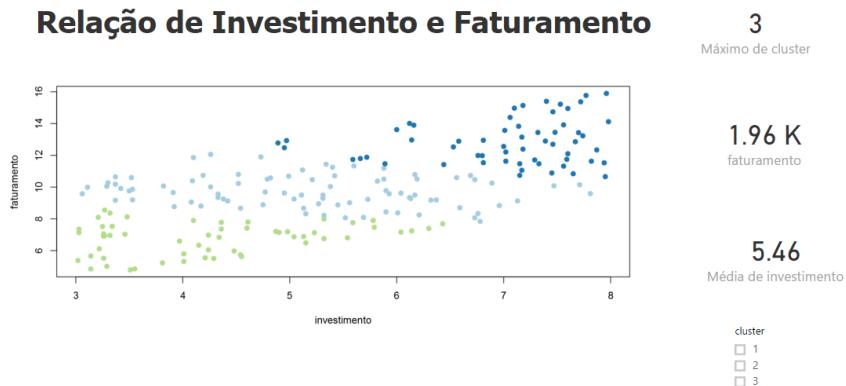


Figura 231 - Gráfico com três clusters

Se por acaso quiser testar e modificar a quantidade de clusters que estes dados devem ser reorganizados, você pode limpar sua tabela de dados do *Azure SQL Database* e em seguida mudar a parametrização da quantidade de clusters que o código em R no módulo do Azure Machine Learning irá gerar.

No código em R informado no *Azure Machine Learning*, a Linha 8 possui a parametrização da quantidade de grupos que o *K-Means* deve montar. Altere de 3 para 10, caso não se lembre onde é, veja a Figura 231 neste capítulo.

Ao executar novamente o experimento, lembrando que sua tabela de dados deve estar vazia (sem nenhum registro), e consultando novamente os dados no Power BI, o resultado esperado é como o apresentado na Figura 232.

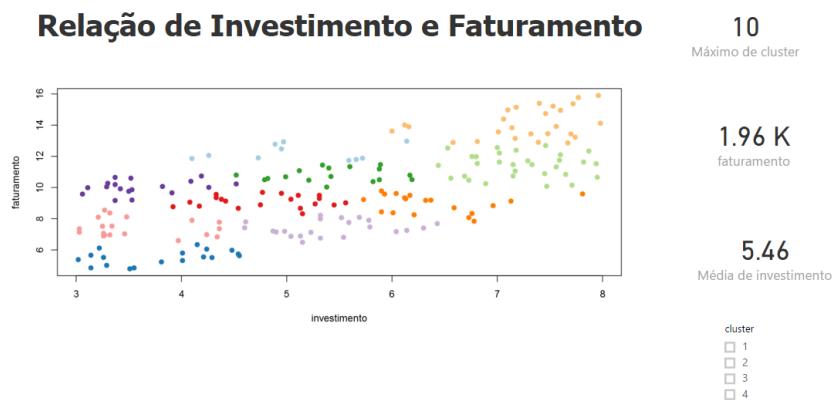


Figura 232 - Gráfico com três clusters

Ao alterar os clusters que você possui no segmentador, você verá que os dados não estão mais somente separados em três grupos, mas sim, em dez. Isso por causa da quantidade de grupos que o algoritmo lá do *Azure Machine Learning* foi capaz de montar.

## Conclusão

Obrigado por chegar até aqui. É uma grande honra para nós que você tenha nos aguentado durante toda a leitura.

Resumindo a leitura, te contamos qual é o papel do cientista de dados no cotidiano das corporações independente do seu tamanho. Comentamos também sobre a importância do estudo de matemática e estatística para aperfeiçoar os algoritmos que são disponíveis nas ferramentas de trabalho, apesar de podermos usar os módulos de Machine Learning que já foram pesquisados e configurados para atender a grande maioria dos cenários empresariais sem a necessidade de modificar nada na questão da equação. Apresentamos uma breve e funcional introdução à Linguagem R para permitir que suas atividades não fiquem limitadas aos módulos existentes na plataforma. Explicamos sobre o desenvolvimento e uso de uma integração através de APIs para que os experimentos pudessem também ser acessados por aplicativos externos. Dentre outras coisas...

A dedicação e o estudo prático aplicado à esta obra com certeza poderão trazer benefícios reais para seus negócios. Aplicando o cenário ideal em uma necessidade real, é possível alcançar resultados que antes eram impensáveis.

Esperamos que com esta produção literária você alcance seus objetivos e possa fornecer benefícios incríveis para sua empresa e clientes.

Forte abraço,

Diego Nogare e Thiago Zavaschi