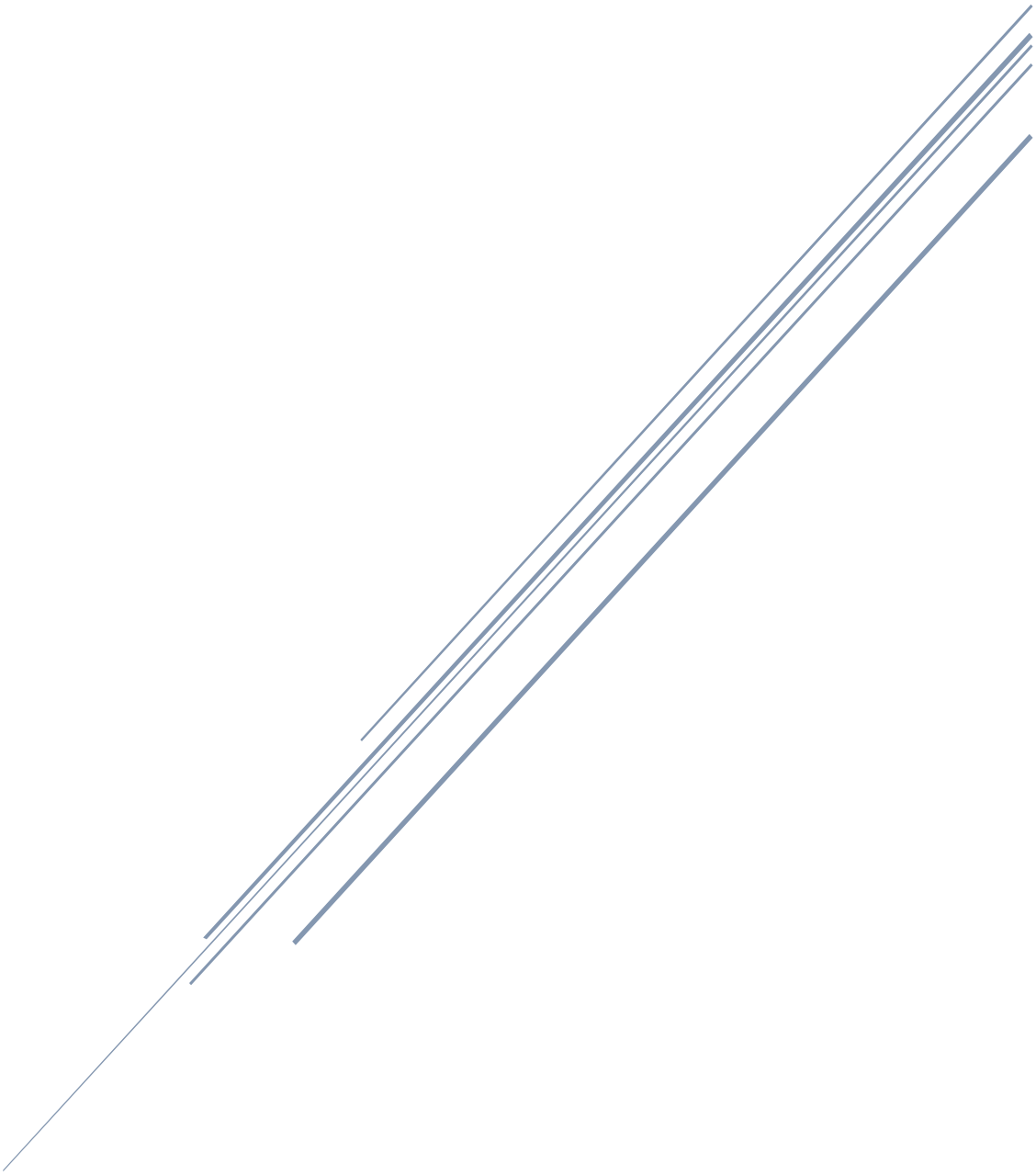
DESAFIO RADIX
MACHINE LEARNING

Processo Seletivo para Engenharia de Dados



Henrique de Souza Santana
UERJ – Engenharia Elétrica / Sistemas Eletrônicos

Sumário

Informações e dicas importantes

• Para resolver o problema, é necessário ter acesso aos arquivos "p1_data_train.csv'', "p1_data_test.csvR e "p1_predictions_example.csv". O link para download se encontra no enunciado do problema abaixo;

• Não é obrigatório apresentar solução completa para o problema, as ideias e criatividade dos candidatos serão valorizadas no processo de avaliação. Quem tiver interesse nas vagas de estágio, e tiver dificuldade em criar a solução, pode explicar em alto nível as suas ideias.

• O objetivo deste desafio é não somente avaliar os candidatos, mas também mostra aos interessados o tipo de problema que os selecionados irão resolver dentro da empresa. Dessa forma, os próprios alunos podem avaliar se tem interesse em trabalhar nessa área;

• No final desse documento, foram passadas referências que podem ajudar na busca de solução para o problema;

• Não há restrições quanto às ferramentas computacionais usadas para resolver o problema. Particularmente, as linguagens R e Python podem ajudar muito na solução. Na verdade, o próprio problema foi criado usando essas linguagens;

• Os candidatos interessados nas vagas devem enviar, além do currículo, um documento pdf com a solução (completa ou parcial) para o problema e também o arquivo p1_predictions.csv", com as predições geradas para o problema;

• O arquivo enviado com predições deve estar no mesmo formato que o arquivo disponibilizado: "p1_predictions_example.csv'';

• Cite no documento pdf com a descrição da solução as ferramentas computacionais utilizadas;

• Na solução pode-se também explicar o que não deu certo em hipotéticas tentativas sem sucesso.

Parte 1 – Problema

Considere um processo industrial que opera em dois estados possíveis de qualidade: ótimo ou regular. A determinação do estado corrente do processo é feita por operadores humanos, que tem acesso a um painel de controle, exibindo uma série de variáveis medidas. Em intervalos regulares, os operadores classificam o estado de qualidade do processo, de acordo com os valores observados das variáveis medidas. Baseado na classificação produzida, ações adequadas são tomadas sobre o processo. Devido a limitações de informatização da planta, nem todas variáveis medidas são armazenadas em um banco de dados histórico.

Você foi contratado para investigar o processo de interpretação realizado pelos operadores. A sua investigação deve ser pautada em análises sobre uma base de dados amostrada do banco de dados histórico. A base ("p1_data_train.csv") a que você tem acesso possui as seguintes variáveis:

• Temp1, Temp2, Temp3 e Temp4, que são temperaturas medidas em diferentes pontos da planta;

• target, que é a variável de resposta, representando o estado de qualidade associado à amostra em questão. Esta variável é binária, com **0** representando o estado ótimo, e **1** o estado regular.

Parte 2 – Desafio

Para concluir sua análise, realize os seguintes passos:

• Faça uma análise exploratória dos dados, criando visualizações para as variáveis do problema;

• Ajuste um modelo de regressão logística para predição da variável de resposta;

• Com base na análise exploratória realizada, e nos coeficientes do modelo paramétrico treinado, enumere quais são as variáveis mais relevantes no processo de interpretação realizado pelos operadores;

• Para avaliar a qualidade da sua modelagem, o seu cliente enviou um conjunto de dados de teste

("p1_data_test.csv"). Utilize o seu modelo paramétrico treinado, para gerar predições para essas amostras de teste. As predições devem estar também no formato binário. Salve as suas predições em um arquivo chamado "p1_predictions.csv";

• Estime qual o erro de suas predições para o conjunto de teste;

• Enumere potenciais motivos de não ser possível ajustar um modelo com erro O para o conjunto de teste que lhe foi enviado.

Parte 3 – Introdução

Machine Learning ou Aprendizado de Máquina, como é normalmente traduzido, é uma sub área da disciplina Inteligência Artificial. Vem ganhando cada vez mais espaço desde sua popularização nos últimos anos: desde previsão de valores de imóveis e sistemas de recomendações de grandes sites de e-commerce ao complexo desafio de fazer máquinas realmente inteligentes.

Das diversas classificações que são consideradas atualmente, duas em particular são as que mais chamam a atenção: *Supervised Learning* e *Unsupervised Learning*. Na primeira, temos um conjunto de dados de entrada que possui uma saída conhecida; a máquina aprende essa correlação e tenta prever as saídas a partir de novas informações. Um exemplo é o filtro de e-mails, onde e-mails novos são classificados como spam ou não spam. Já em Unsupervised Learning, temos um conjunto de dados onde não se sabe nenhuma informação a respeito. Aqui, a máquina pode tentar agrupar em conjuntos com características similares.

Dos algoritmos mais usuais para Supervised Learning, temos as funções lineares: *Linear Regression* e *Logistic Regression*, ou em português: Regressão Linear e Regressão Logística. No primeiro caso, o modelo é uma função do tipo y = ax + b para regressão simples com uma variável. Levando em consideração o caso da previsão de valor de um imóvel, y poderia ser o 'valor do imóvel' e x o atributo 'área do imóvel'. Já na Classificação Logística, a ideia é prever valores booleanos ou categóricos como tumor maligno ou benigno baseado em atributos como tamanho do tumor.

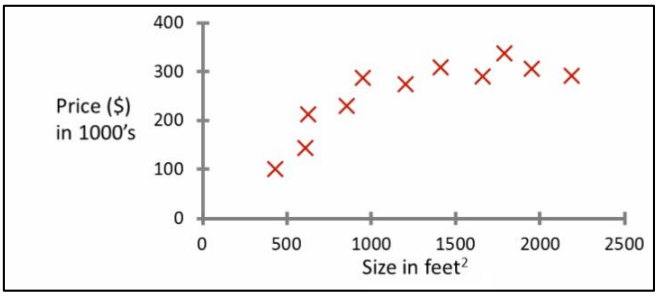


Figura 1 - Exemplo de Regressão Linear

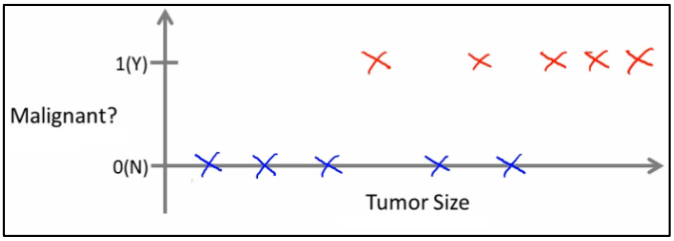


Figura 2 - Exemplo de Regressão Logística

Como o desafio propõe um problema de classificação, a resolução será dada com o algoritmo de Regressão Logística.

Parte 4 – Logistic Regression

- **Intuição**

Quando tratamos de problemas de classificação, o nosso modelo ou função de previsão (ou predição) é chamada de *hypothesis*. Para regressão linear, a hypothesis é $h_\theta(x) = (\theta^T x)$, onde θ é um vetor de parâmetros (ax + b seria $\theta_1 x + \theta_0$). Para regressão logística, usando a função sigmoid $g(z) = 1/(1 + e^{-z})$ em $h_\theta(x) = g((\theta^T x))$, temos a hypothesis como:

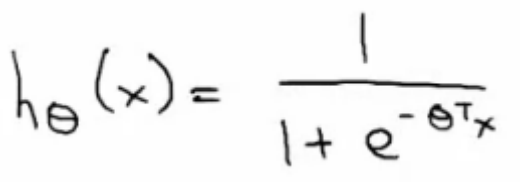


Figura 3 - hypothesis para Regressão Logística

E a função sigmoid é apresentada na Figura 4:

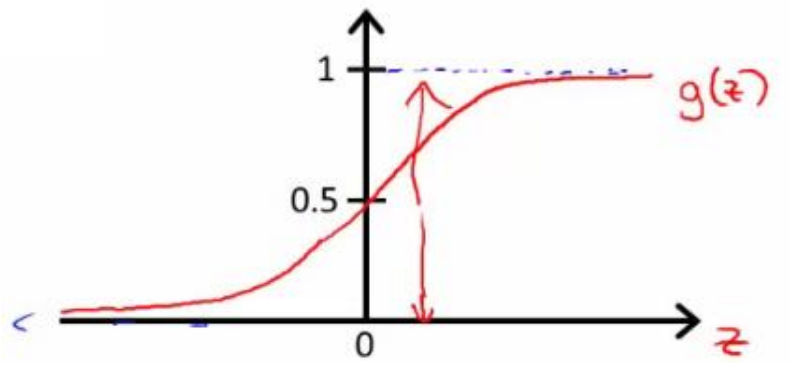


Figura 4 - Sigmoid Function

A intuição para utilizar o modelo de Regressão Logística ao invés do modelo de Regressão Linear pode ser observada na figura 5. Uma linha reta não seria muito útil para classificação ou previsão de probabilidade de um evento ocorrer ou não, por exemplo. A não ser que fosse 'dobrada' em seus extremos.

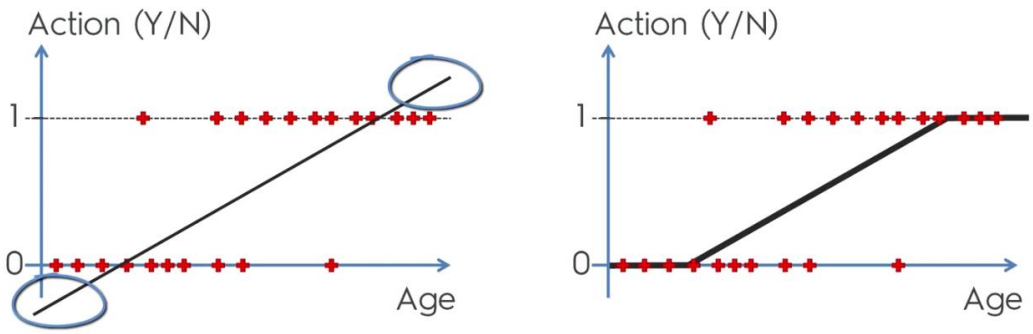


Figura 5 - Intuição para Regressão Logística

- **Decision Boundary**

Outro termo importante para a Classificação é a *Decision Boundary* (ou fronteira de decisão). É uma linha (ou superfície, em dimensões superiores) que separa duas classes (sim ou não, 0 ou 1, ótimo ou regular).

Essa fronteira dá um melhor entendimento de como a hypothesis se parece.

Assim podemos usar a função sigmoid da seguinte maneira:

- Se a probabilidade de ser 1 é maior do que 0.5, então a predição de y = 1;
- Se não, predição de y = 0.

Olhando novamente para a função sigmoid:

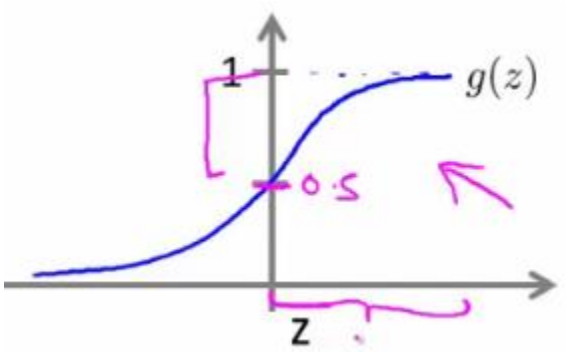


Figura 6 - Entendendo a função Sigmoid

A função g(z) é maior ou igual a 0.5 quando z é maior ou igual a 0. Em outras palavras, se z é positivo, então g(z) é maior do que 0.5. Lembrando que $z = (\theta^T x)$.

Então quando $\theta^T x >= 0$, temos nossa hypothesis $h_\theta >= 0.5$

Se por exemplo, $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ com $\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

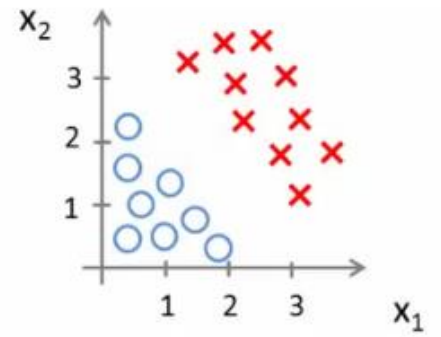


Figura 7 - Exemplo de Regressão Logística

Então, nosso vetor de parâmetros é um vetor coluna com os valores acima. A transposta é um vetor linha: θ^T = [-3, 1, 1] e z se torna $\theta^T x$.

A predição será y = 1 se:

$$-3x_0 + 1x_1 + 1x_2 >= 0$$
$$-3 + x_1 + x_2 >= 0$$

Reescrevendo, se $(x_1 + x_2 >= 3)$ se então predizemos que y = 1.

Se plotarmos $x_1 + x_2$ = 3, teremos graficamente nossa decision boundary.

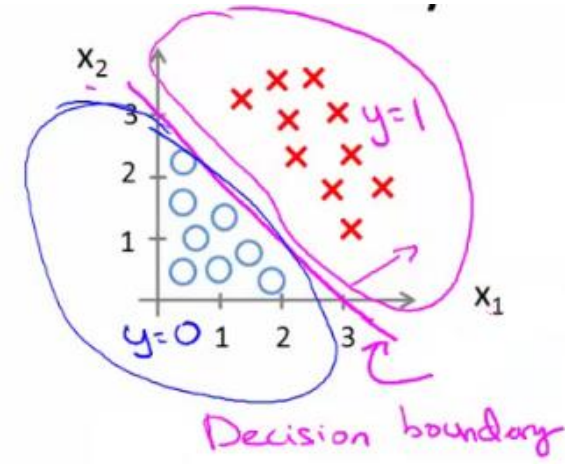


Figura 8 - Plotando a Decision Boundary

- **Cost function e Minimização para Logistic Regression**

Cost function ou Função de Perda é uma função que mapeia um evento ou valores de uma ou mais variáveis num número real intuitivamente apresentando algum custo associado ao evento. Um problema de otimização (minimização) procura minimizar uma função de perda.

Em outras palavras, a Função de Perda nos permite descobrir quão bem uma linha reta se adequa aos nossos dados.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Figura 9 - Fórmula da Cost Function para Logistic Regression

- **Gradient Descent**

O Método do Gradiente como também é conhecido é usado para encontrar o mínimo local de uma função usando um esquema iterativo, onde cada passo se toma a direção do gradiente, que corresponde a direção de declive máximo. Podemos dizer que encontraremos o mínimo quando as derivadas parciais em relação a θ_0 e θ_1 forem iguais a 0. A fórmula da Figura 10 vale tanto para Linear Regression quanto para Logistic regression.

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Figura 10 - Fórmula para Gradient Descent

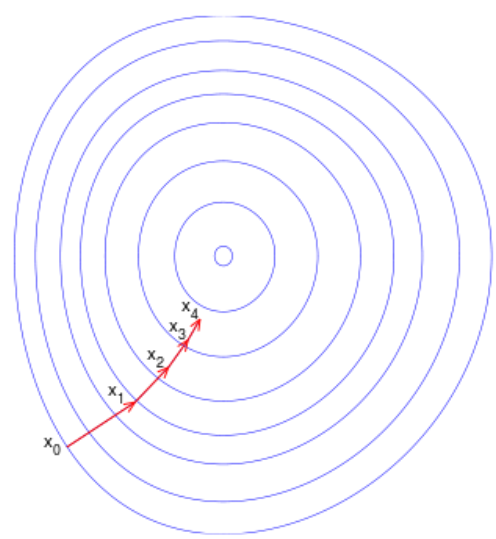


Figura 11 - Intuição para Gradient Descent

Parte 5 – Ferramentas Computacionais Usadas no Desafio

A princípio, o desafio começou a ser resolvido em Matlab/Octave. Entretanto, a praticidade da linguagem Python com todas as suas bibliotecas e o crescente uso de pacotes como pandas fizeram com que a solução fosse extraída com o uso de Python. Além, é claro, da sugestão no próprio documento de apresentação do desafio.

O desafio foi feito na plataforma Anaconda da Continuum. Utilizando bibliotecas como pandas, numpy, scikit learn e matplot lib. Como IDE, o Spyder foi utilizado.

Parte 6 – Visualizando os dados

Para gerar os scatter plots da figura 12 foi utilizado a função scatter_matrix do módulo pandas.tools.plotting.

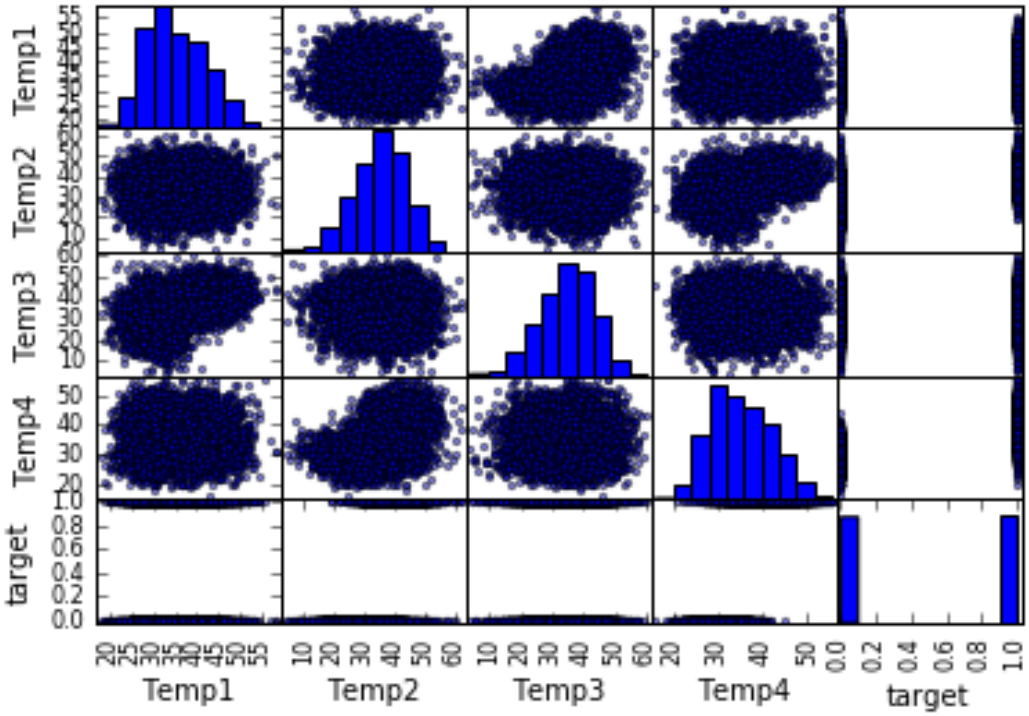


Figura 12 - Matriz com scatter plots

Para gerar os histogramas foi utilizado pandas.DataFrame.hist().

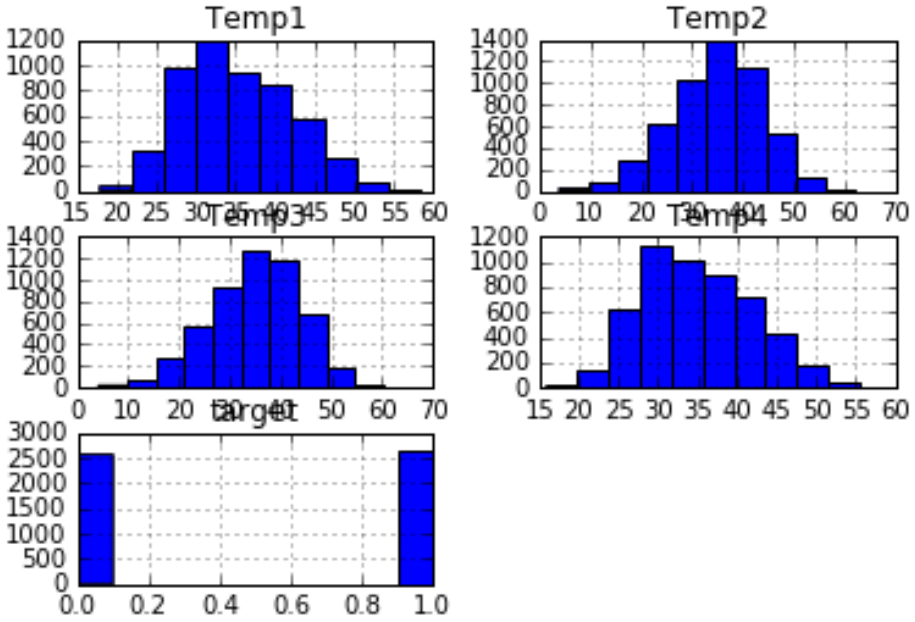


Figura 13 - Histogramas

As imagens foram geradas usando plt.show() e extraídas diretamente do IPython Console.

Parte 7 – Solução do Desafio

Como mencionado, a primeira versão da solução do Desafio estava sendo desenvolvida em Matlab. Porém, desenvolvendo em python e utilizando scikit learn possibilitou que não houvesse a necessidade de se preocupar com Cost Function ou Grandient Descent.

Foi gerado um arquivo chamado solution_henrique.py contendo as informações do código (sem os plots). Entretanto, apresento o mesmo na Parte 7.

Considerações:

- dataset é conjunto de dados inicial;

- customer_dataset é o conjunto de dados enviado pelo cliente para teste;

- O arquivo p1_predictions.csv contém o valor de y com as predições para o conjunto de dados enviado pelo cliente (customer_dataset). O arquivo foi gerado a partir do código e em seguida editado removendo a coluna de índice;

- Como customer_dataset não possuía a y (0 ou 1), no código foi usado 20% dos dados do dataset inicial (p1_data_train) para teste e verificação das métricas.

Métricas

- Matriz de Confusão

	0	1
0	464	54
1	39	485

Figura 14 - Confusion Matrix (cm)

- Precisão (ac) = 0.91074856046065256

- Relatório de Classificação

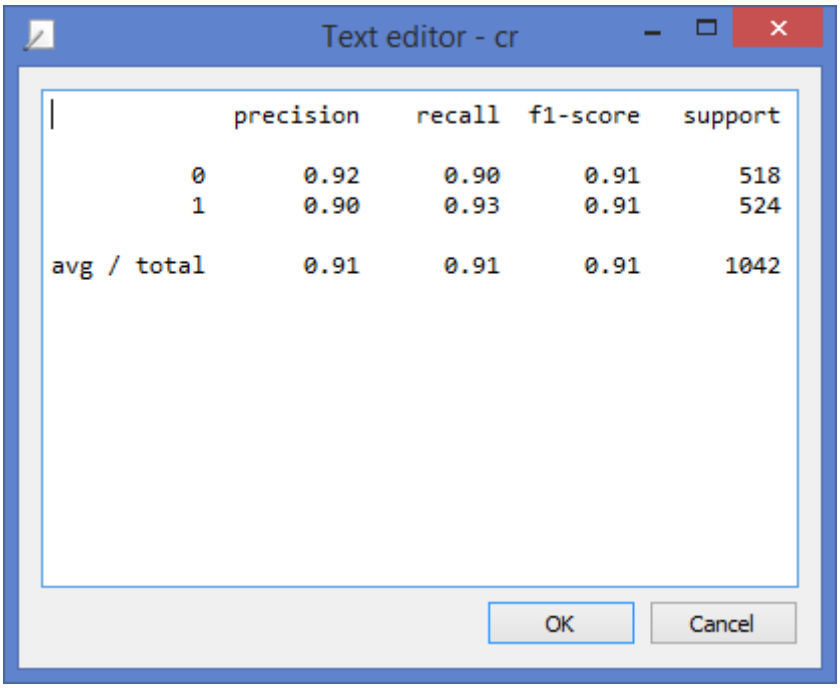


Figura 15 - Classification Report (cr)

Visualização das Variáveis

Name	Type	Size	Value
X	float64	(5209, 4)	array([[37.86132868, 19.50045514, 36.65075913, 29… [43.36886385, 33.65170925, 35.36361147, 34…
X_customer	float64	(5208, 4)	array([[44.76023175, 30.15548245, 42.15333737, 29… [31.25787078, 30.21395992, 24.585232 , 34…
X_test	float64	(1042, 4)	array([[30.0920832 , 44.55937833, 33.65112781, 36… [30.31906827, 40.00971759, 21.39943767, 38…
X_train	float64	(4167, 4)	array([[32.63636137, 8.81100275, 33.2129605 , 34… [35.06600165, 44.7016839 , 33.26116628, 26…
ac	float64	1	0.91074856046065256
cm	int32	(2, 2)	array([[464, 54], [39, 485]])
cr	str	1	precision recall f1-score support
customer_dataset	DataFrame	(5208, 4)	Column names: Temp1, Temp2, Temp3, Temp4
dataset	DataFrame	(5209, 5)	Column names: Temp1, Temp2, Temp3, Temp4, target
y	int64	(5209,)	array([0, 1, 1, ..., 1, 1, 1], dtype=int64)
y_pred	int64	(1042,)	array([1, 1, 1, ..., 0, 0, 0], dtype=int64)
y_test	int64	(1042,)	array([1, 1, 1, ..., 0, 0, 0], dtype=int64)
y_train	int64	(4167,)	array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

Figura 16 - Variable Explorer

Parte 8 – Código

```
# Desafio Radix usando Logistic Regression

# Importando as bibliotecas
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importando o dataset para treinamento e teste
dataset = pd.read_csv('p1_data_train.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values

# Importando o dataset do cliente
customer_dataset = pd.read_csv('p1_data_test.csv')
X_customer = customer_dataset.iloc[:, :].values

# Dividindo the dataset em Training set e Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# Ajustando o modelo Logistic Regression ao Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

# Predição dos resultados do Test set
y_pred = classifier.predict(X_test)

# Avaliando a performance do algoritmo
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
cm = confusion_matrix(y_pred, y_test)
ac = accuracy_score(y_pred, y_test)
cr = classification_report(y_pred, y_test)

# Predição dos resultados do dataset do cliente e solução
y_pred_customer_test = classifier.predict(X_customer)
df = pd.DataFrame({'target': y_pred_customer_test})
df.to_csv('p1_predictions.csv')
```

Parte 9 – Conclusão

Apesar de haver outros algoritmos de classificação como KNN, SVM e NB, o modelo com Logistic Regression dá conta do recado para esse desafio, com precisão de 91%. Podemos estimar, então, um erro de 9% para o conjunto de dados enviado pelo cliente.

Não é possível ajustar um modelo com erro 0. No caso do Logistic Regression, temos uma linha reta. No exemplo da figura 17, alguns valores de y = 0 estão após a fronteira e vice-versa.

Porém, se escolhêssemos um modelos com polinômios de graus mais elevados, poderíamos cair no problema de *overfitting*, que é ter um modelo que só se adapta ao training_set e falha quando recebe novos dados para predição.

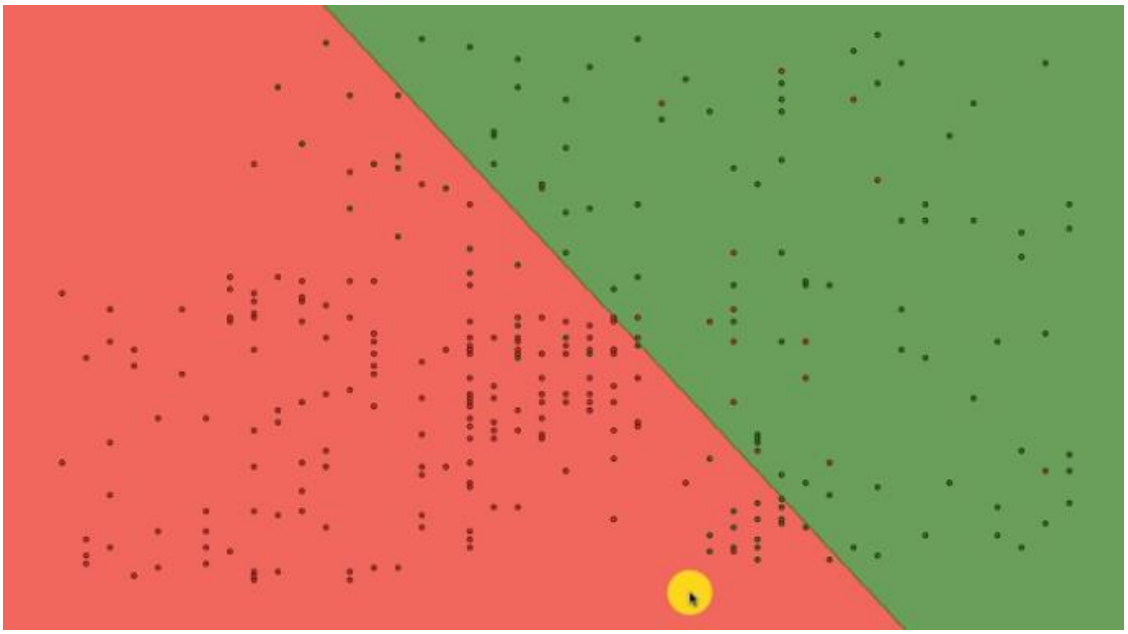


Figura 17 - Decision Boundary no Python

Por fim, com uma rápida visualização dos dados, percebe-se que as variáveis Temp4, Temp2 e Temp1 são as mais relevantes no processo de interpretação. Não foram utilizadas técnicas como PCA na análise.

Parte 10 – Bibliografia

- Russel, Stuart & **Norvig**, Peter. Inteligência Artificial. EditoraCampus, Terceira Edição;

- Machine Learning – Stanford University (https://www.coursera.org/)

- Machine Learning A-Z™: Hands-On Python & R In Data Science (https://www.udemy.com/)

- Wikipedia (https://en.wikipedia.org)

- Machine Learning Mastery (http://machinelearningmastery.com/)