Ministério da Educação

**Universidade Tecnológica Federal do Paraná**

Câmpus Medianeira Ciência da Computação

**Biblioteca Scikit-learn**

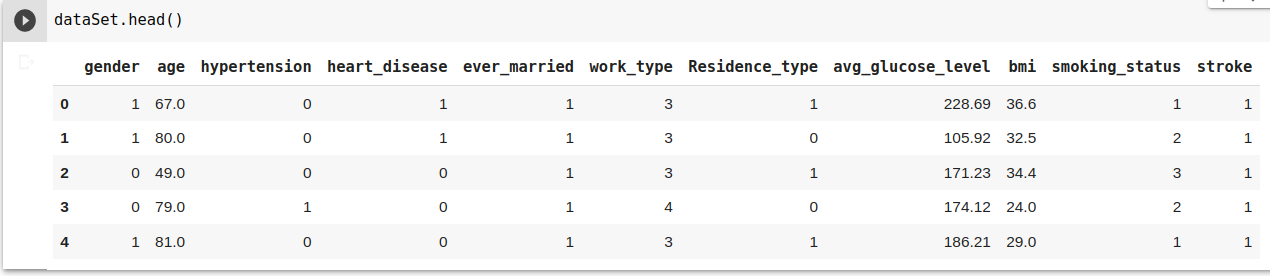
Emanuel Mello Nogueira de Oliveira

01 de novembro de 2022

# DATASET

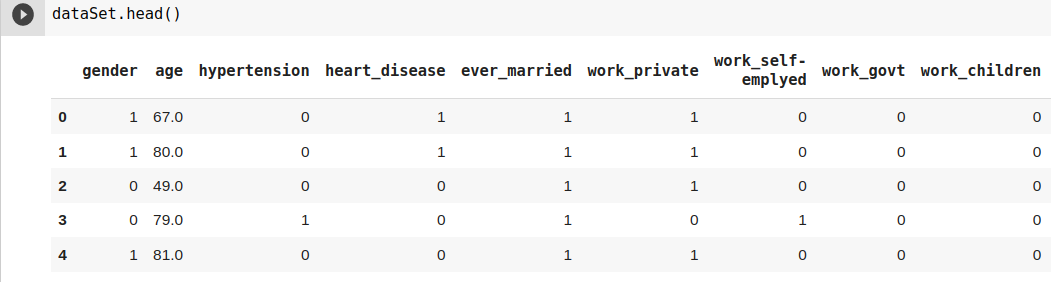
Dataset utilizado foi o brain\_stroke.csv, um dataset utilizado para comparar dados relacionados a AVC, contendo os seguintes atributos: gender, age, hypertension, heart disease, ever married, work type, avg glucose level, bmi, smoking status, stroke. Cada um desses atributos contendo diferentes tipos de dados que foram formatados em dois datasets.

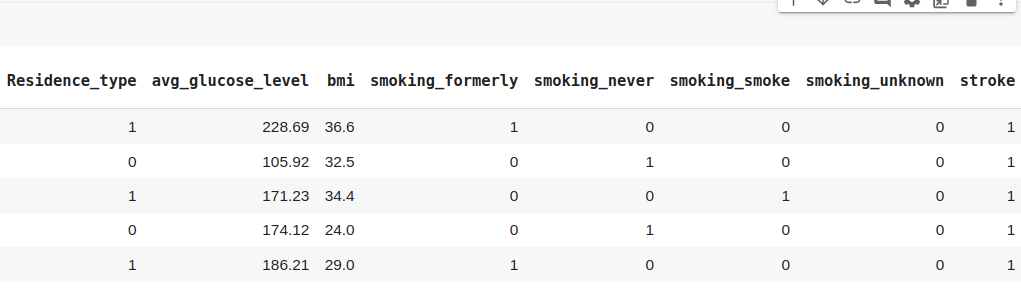
Brain\_stroke\_formated.csv que altera os dados para dados numericos, como gender: 1 macho, 0 femea, ever married: 1 sim, 0 não, work type: 1 child, 2 govt, 3 private, 4 self-employed, residence type: urban 1, rural 0, smoking status: formerly 1, never 2, smoke 3, unknow 4.



# *Representação do dataset Brain\_stroke\_formated.csv.*

Brain\_stroke\_formated\_separated.csv que separa algumas colunas em mais de uma atribuindo 1 caso possua esse atributo e 0 caso não possua, como work type em work\_private, work\_self-employed, work\_govt, work\_children, e smoking satus em smoking\_formerly, smoking\_never, smoking\_smoke, smoking\_unknown.





# *Representação do dataset Brain\_stroke\_formated\_separated.csv.*

# SCIKIT-LEARN

Fora utilizado o Scikit-learn que é uma biblioteca Python construída especificamente para aplicação prática de aprendizado de máquina. Essa biblioteca possui uma série de ferramentas que são de fácil uso e muito eficientes para realizar a análise preditiva dos dados.

# CLASSIFICADORES

# 3.1 Arvore de Decisão

# Arvores de decisão são métodos supervisionados não paramétricos usados para classificação e regressão, o objetivo e criar um modelo que prediz o valor da variável alvo ao aprender decisões simples através de dados.

# Fora utilizado o dataset “Brain\_stroke\_formated.csv” para o treinamento, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.2 **Support Vector Machine – kernel linear**

# SVMs são um set de treinamento supervisionado usado para classificação, regressão e detecção de outliers, algumas vantagens de utilizar maquinas de vetor suporte são, efetividade em altas dimensoes, efetivo em casos que o numero de dimensões e maior que o numero de samples, além de ser bem versátil pois e possível utilizar outras funções de kernel.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, foi utilizado o kernel linear, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.3 **Support Vector Machine - kernel rbf**

# SVMs são um set de treinamento supervisionado usado para classificaca, regressao e deteccao de outliers, algumas vantagens de utilizar maquinas de vetor suporte são, efetividade em altas dimensoes, efetivo em casos que o numero de dimensoes e maior que o numero de samples, alem de ser bem versatil pois e possível utilizar outras funcoes de kernel.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, foi utilizado o kernel rbf, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.4 **K Nearest Neighbors - k= 3**

# O principio por tras de nearest neighbors e encontrar um numero predefinido de amostras a uma distancia próximas do novo ponto, e predizer o label atraves deste ponto, o numero de amostras pode ser uma uma constante pre definida, ou variavel baseado na densidade local dos pontos.A distancia pode em geral, ser qualquer medida metrica como distancia euclidiana.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, foi utilizado o raio de 3 pontos, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.5 **K Nearest Neighbors – k = 6**

# O principio por tras de nearest neighbors e encontrar um numero predefinido de amostras a uma distancia próximas do novo ponto, e predizer o label atraves deste ponto, o numero de amostras pode ser uma uma constante pre definida, ou variavel baseado na densidade local dos pontos.A distancia pode em geral, ser qualquer medida metrica como distancia euclidiana.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, foi utilizado o raio de 6 pontos, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.6 **K Nearest Neighbors – p = 1**

# O principio por tras de nearest neighbors e encontrar um numero predefinido de amostras a uma distancia próximas do novo ponto, e predizer o label atraves deste ponto, o numero de amostras pode ser uma uma constante pre definida, ou variavel baseado na densidade local dos pontos.A distancia pode em geral, ser qualquer medida metrica como distancia euclidiana.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, foi utilizado o raio de 6 pontos, e utilizando distancia manhattan com p=1, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.7 **Gaussian Naive Bayes**

# Naive bayes gaussiano e um modelo generativo, ele assume que cada classe segue uma distribuição gaussiana, a diferença entre um QDA e um Naive Bayes Gaussiano e que o naive bayes pressupõe a independencia de features, significando que as matrizes de covariância são matrizes diagonais, alem de possuir matrizes de variâncias que são especificas por classe.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, foi utilizado a “variavel random\_state” com valor “0”, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.8 **Categorical Naive Bayes**

# Naive bayes categorico e utilizado para classificacao com features discretos/distintos que são categoricamente distribuidos, a categoria de cada feature e extraida atraves de uma distribuicao categorica.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*

# 3.9 **Linear Regression**

# Regressao Linear e um metodo para modelar a relacao entre medidas escalares de uma ou mais variaveis, as relacoes são modelas usando funcoes de predicao linear modelando parametros atravez de dados, assim como outras formas de regressao, a regressao linear foca na distribuicao de probabilidade condicional dado os valores das predicoes, ao inves de distribuicao de probabilidade conjunta dessas variaveis.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Dado que a função de acurácia e uma métrica de classificação, não e possível utilizá-la em regressões, então apenas foi analisado o score de cada X e y em relação as variáveis de treino e teste, e o calculo de uma loss usando numpy.

# 

# *Representação do calculo de loss, e score.*

# Posteriormente fora utilizado o dataset “Brain\_stroke\_formated\_separated .csv” para o treinamento, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# 

# *Representação do calculo de loss, e score.*

# 3.10 **Perceptron**

# O perceptron e um algoritmo de aprendizado supervisionado de classificadores binarios, um classificador binario seria uma funcao que decide se um input, representado por um vetor de numeros, pertence ou não a uma classe especifica, sendo este um tipo de classificador linear, realizando predicoes baseados em uma funcao de predicao linear, combinando pesos com features.

# Fora utilizado o dataset “Brain\_stroke\_formated .csv” para o treinamento, os dados do dataset foram divididos entre 20% para teste e 80% para o treinamento, mais precisamente entre variáveis como X\_train, X\_test, y\_train, y\_test, as variáveis de X\_train, e Y\_train foram utilizadas na função de fitting “clf.fit(X\_train, y\_train)”.

# Utilizando a função de accuracy\_score foi possível observar a precisao do algoritmo utilizado.

# 

# *Representação do calculo de acurácia.*