Sistemas Inteligentes - IF684

Prof: Adriano Lorena

Projeto Aprendizado Supervisionado

Integrantes

Lucas Henrique - lhns3

Introdução

Objetivo de limpar, analisar, explorar os dados e utilizar modelos de classificação nos dados tratados, utilizando os melhores parâmetros e interpretando e discutindo os resultados.

O dataset escolhido foi o Breast Cancer Wisconsin (Diagnostic), que conta com dados referentes aos núcleos celulares de massa mamária e o diagnóstico (**target**) de câncer benigno ou maligno dos pacientes.

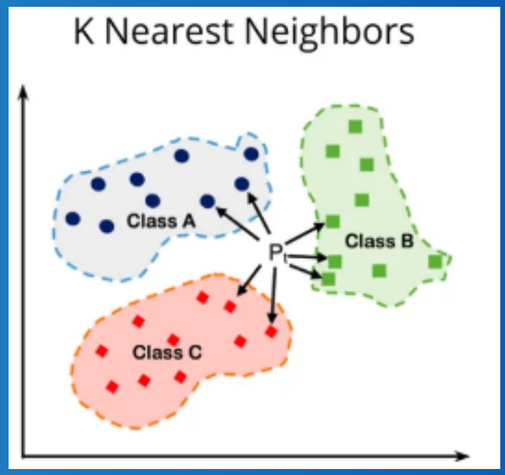
- Características do Dataset: **Multivariado**
- Área: **Ciência da Vida**
- Tarefas associadas: **Classificação**
- Tipo dos atributos: **Real**
- Instâncias: **569**
- Atributos: **32**

***Observações:**
ID - > inteiro
Diagnóstico -> objeto
B -> benigno
M -> maligno
10 atributos com a média, erro padrão e o pior

Fundamentos

kNN (k-Nearest-Neighbors)

- armazena todos os casos possíveis
- sistema de votos de maioria
- parâmetro k (geralmente 3-10)
- calculado a partir de distâncias
- k não pode ser múltiplo do número de classes
- simples, eficiente, modelo não entendível e lento na classificação



Fundamentos

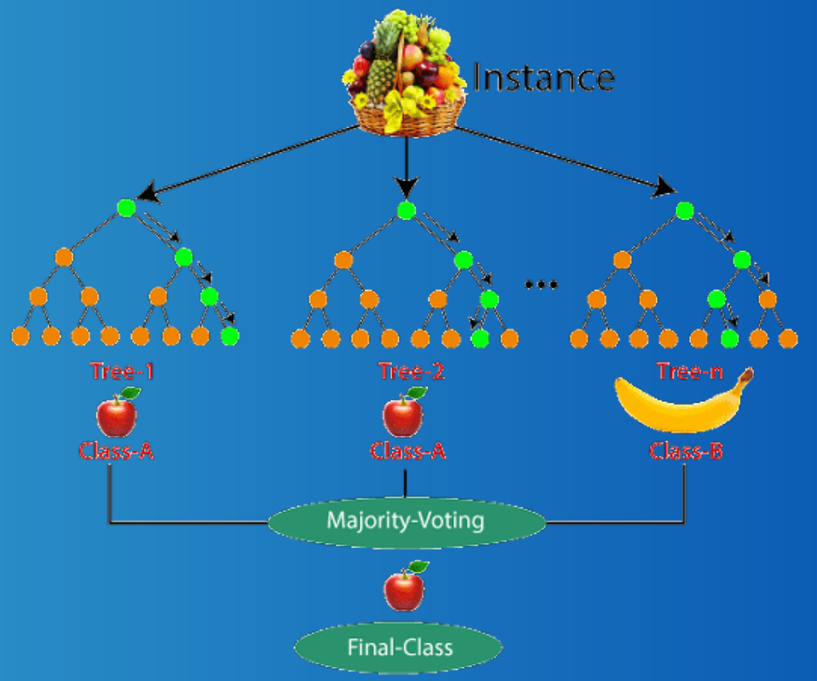
Árvores de decisão

- pode-se pensar em um arranjo de perguntas que levam a uma decisão
- estrutura de árvore semelhante a fluxograma hierárquico
- nós internos é um atributo, um ramo uma decisão e um nó folha a saída
- algoritmo “caixa branca”, fácil entendimento
- encontra atributos que recursivamente bifurca a árvore a partir do ASM
- Information Gain, Gain Ratio e Gini Index
- captura padrões não-lineares facilmente
- requer menos pré-processamento e é não-paramétrico
- sensitivo a dados com ruídos, pequenas alterações nos dados e dataset desequilibrado
- pouco robusto a muitas dimensões

Fundamentos

Random Forest

- Conjunto de árvores de decisão
- Diversas árvores criadas com diferentes subconjuntos do dataset
- Resultado é o voto mais popular ou a média das n árvores
- Tem uma ótima precisão
- Lida bem com alto volume de dados e variáveis



Metodologia

Visualização inicial dos dados

- 33 colunas
- 569 instâncias
- última coluna com números indefinidos
- apenas **diagnosis** é categórica

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 33 columns

```
] df.shape
```

```
(569, 33)
```

symmetry_worst	fractal_dimension_worst	Unnamed: 32
0.4601	0.11890	NaN
0.2750	0.08902	NaN
0.3613	0.08758	NaN
0.6638	0.17300	NaN
0.2364	0.07678	NaN

```
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
```

Metodologia

Pré-Processamento

- Retirada das colunas id e Unnamed
- Id não tem relação com o diagnóstico
- Unnamed tem valores indefinidos

```
[159] # Retirando a coluna de id e Unnamed
      df.drop('id',axis=1,inplace=True)
      df.drop('Unnamed: 32',axis=1,inplace=True)
```

- Trocamos M e B por 1 e 0

```
df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0})
df.head()
```

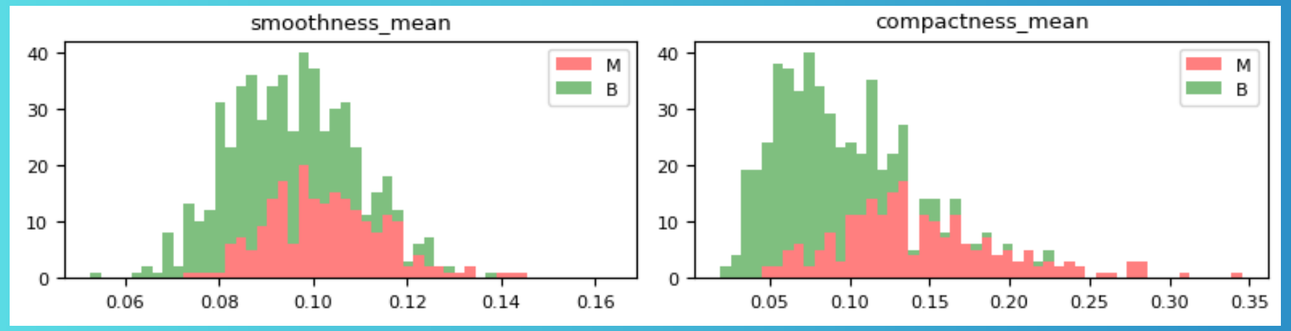
	diagnosis	radius_mean	texture_mean	perimeter_mean
0	1	17.99	10.38	122.80
1	1	20.57	17.77	132.90
2	1	19.69	21.25	130.00
3	1	11.42	20.38	77.58
4	1	20.29	14.34	135.10

5 rows × 31 columns

Metodologia

Exploração dos dados

- Algumas variáveis tem relação direta com tumores malignos ou benignos
- À medida que compactness aumenta percebe-se mais tumores malignos
- Smoothness parece não influir no tipo do tumor




```
[ ]  # Empilhando os dados
     plt.rcParams.update({'font.size': 8})
     fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(8,10))
     axes = axes.ravel()
     for idx,ax in enumerate(axes):
         ax.figure
         binwidth= (max(df[features_mean[idx]]) - min(df[features_mean[idx]]))/50
         ax.hist([dfM[features_mean[idx]],dfB[features_mean[idx]]], bins=np.arange
         ax.legend(loc='upper right')
         ax.set_title(features_mean[idx])
     plt.tight_layout()
     plt.show()

     # percebe-se que quando radius, perimeter, area, compactness, concavity, conc
     # enquanto texture, smoothness, symmetry e fractal_dimension médios parecem n
```

Metodologia

Divisão e Normalização (kNN)

- Normalização dos valores no kNN, por ser um algoritmo que utiliza do cálculo de distâncias
- Divisão de 20% dos dados para teste e 80% para treino

```
# Divisão dos dados em atributos (X) e target (Y)
X = df.drop('diagnosis', axis=1)
y = df['diagnosis']

# Divide os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Metodologia

Parâmetros arbitrários

Modelo	Acurácia padrão
kNN	96%
AS	94%
RF	94%

```
[ ] knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(X_train, y_train)

    KNeighborsClassifier
    KNeighborsClassifier(n_neighbors=3)

    y_pred = knn.predict(X_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    print("Acurácia:", accuracy)

    Acurácia: 0.9649122807017544
```

```
[ ] # Criação da árvore de decisão
    clf = DecisionTreeClassifier()

    # treinamento do classificador com nossos dados
    clf = clf.fit(X_train,y_train)

    # Predição da resposta para o dataset
    y_pred = clf.predict(X_test)

[ ] # Testando a acurácia da árvore de decisão
    print("Acurácia:",metrics.accuracy_score(y_test, y_pred))

    Acurácia: 0.9385964912280702
```

```
[ ] rf = RandomForestClassifier()
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    print("Acurácia:", accuracy)

    Acurácia: 0.9385964912280702
```

Metodologia

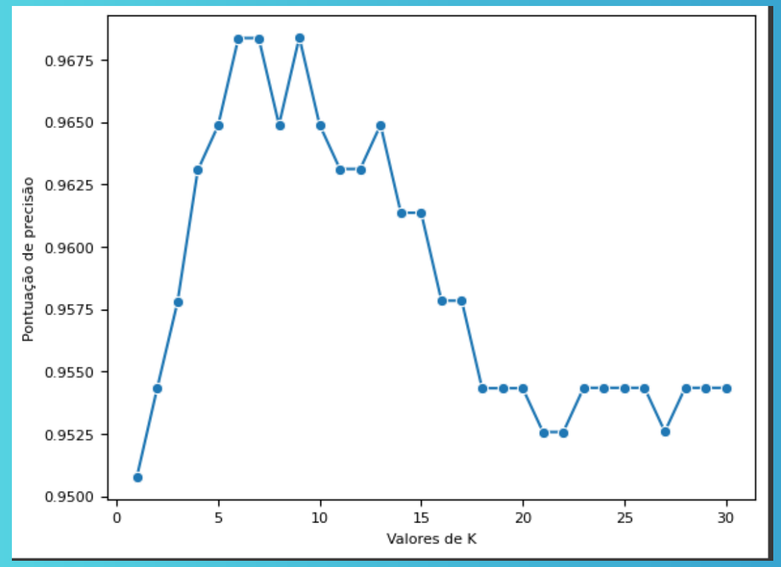
Escolha de parâmetros

Modelo	Acurácia padrão	Método
kNN	96%	Cross-Validation
AS	94%	Mudança na profundida máxima e entropia
RF	94%	RandomizedSearchCV

Resultados

kNN

Cross Validation Score




```
[ ]  k_values = [i for i in range (1,31)]
     scores = []

     scaler = StandardScaler()
     X = scaler.fit_transform(X)

     for k in k_values:
         knn = KNeighborsClassifier(n_neighbors=k)
         score = cross_val_score(knn, X, y, cv=5)
         scores.append(np.mean(score))
         score_mean = (np.mean(score))*100
         print(f'Para o valor {k} de k obtemos {score_mean}% de Cross Validation Score')
```

K = 3 : 96% de Acurácia

```
Acurácia: 0.9824561403508771
Precisão: 1.0
Recall: 0.9487179487179487
```

K = 9

Resultados

Árvore de decisão

entropia e max_depth variável

```
[ ]  for i in range(1,11):
       # Criação da árvore de decisão
       clf = DecisionTreeClassifier(criterion="entropy", max_depth=i)

       # treinamento do classificador com nossos dados
       clf = clf.fit(X_train,y_train)

       # Predição da resposta para o dataset
       y_pred = clf.predict(X_test)

       # Testando a acurácia
       print("Accuracy:",metrics.accuracy_score(y_test, y_pred),"para máximo de profundidade = ", i)
```

```
Accuracy: 0.868421052631579 para máximo de profundidade =  1
Accuracy: 0.868421052631579 para máximo de profundidade =  2
Accuracy: 0.9122807017543859 para máximo de profundidade =  3
Accuracy: 0.9649122807017544 para máximo de profundidade =  4
Accuracy: 0.9473684210526315 para máximo de profundidade =  5
Accuracy: 0.9473684210526315 para máximo de profundidade =  6
Accuracy: 0.9473684210526315 para máximo de profundidade =  7
Accuracy: 0.9385964912280702 para máximo de profundidade =  8
Accuracy: 0.9385964912280702 para máximo de profundidade =  9
Accuracy: 0.956140350877193 para máximo de profundidade =  10
```

max_depth = 6 e gini :

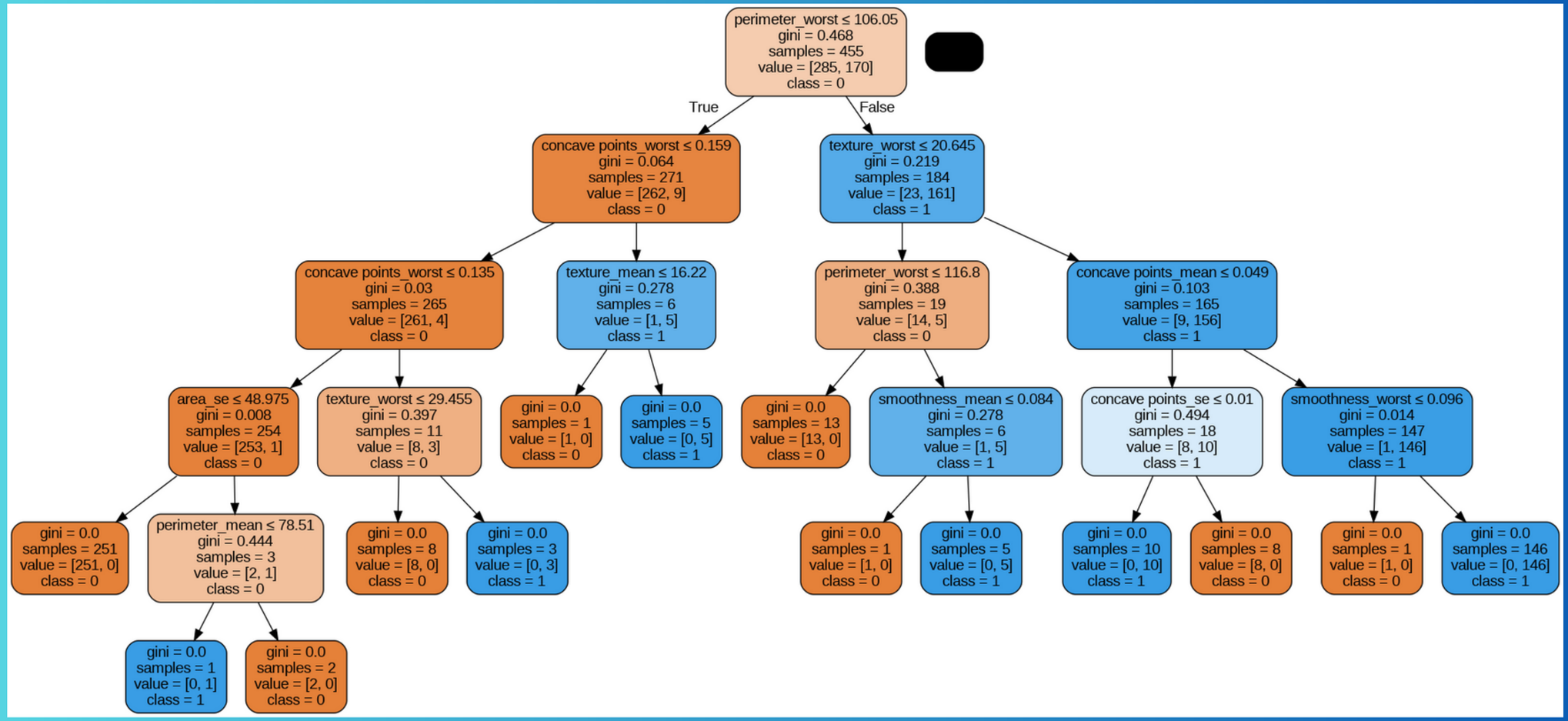
94% de Acurácia

```
Acurácia: 0.9649122807017544
Precisão: 1.0
Recall: 0.9047619047619048
```

entropia e max_depth = 4

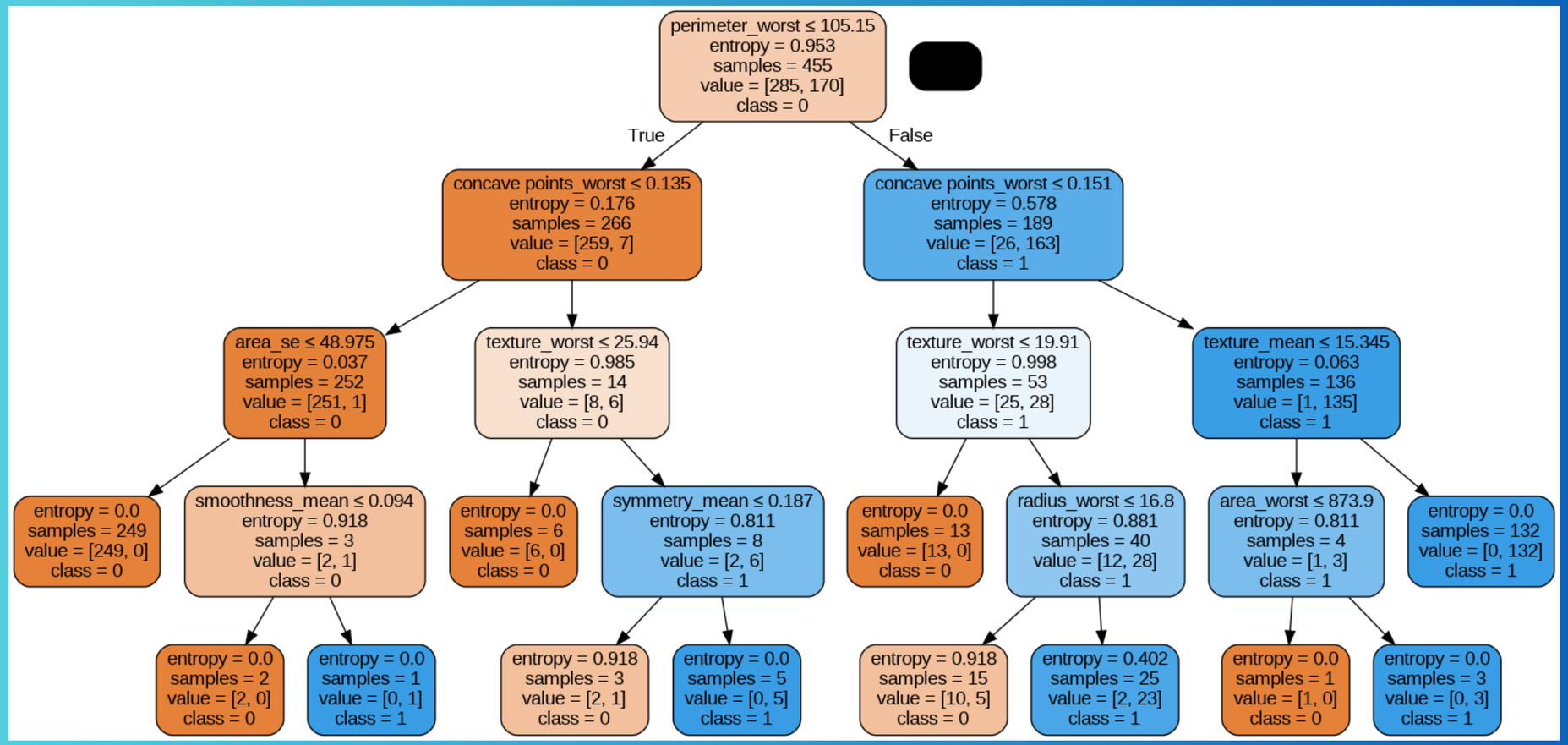
Resultados

Árvore de decisão



Resultados

Árvore de decisão



Resultados

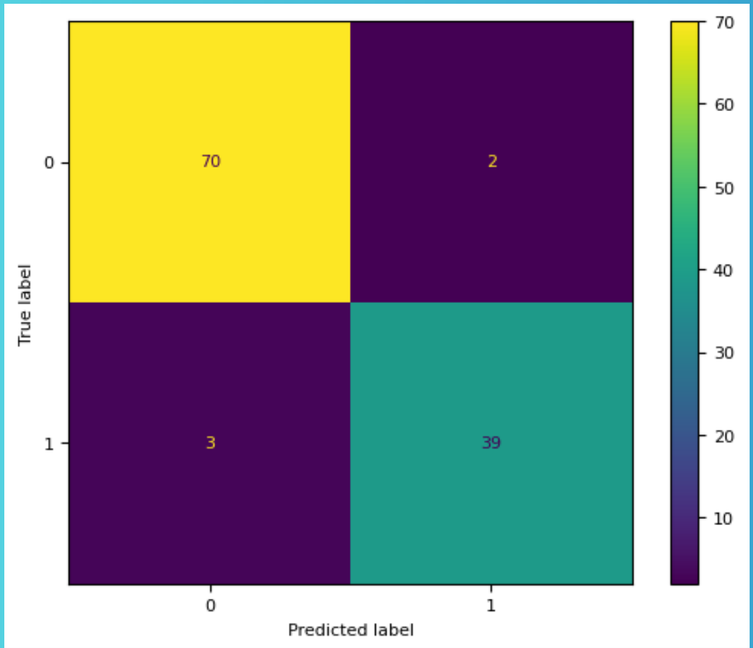
Random Forest

```
[ ]  param_dist = {'n_estimators': [randint(50, 500) for _ in range(4)],
                   'max_depth': [randint(1, 20) for _ in range(4)]}

     # Criando uma random forest
     rf = RandomForestClassifier()

     # Uso de random search para melhorar os parâmetros
     rand_search = RandomizedSearchCV(rf,
                                      param_distributions = param_dist,
                                      n_iter=5,
                                      cv=5)

     # Fit the random search object to the data
     rand_search.fit(X_train, y_train)
```



n_estimators = 3 : 94%

de Acurácia

```
Acurácia: 0.9561403508771193
Precisão: 0.9512195121951219
Recall: 0.9285714285714286
```

n_estimators = 303 e max_depth = 5

Conclusões

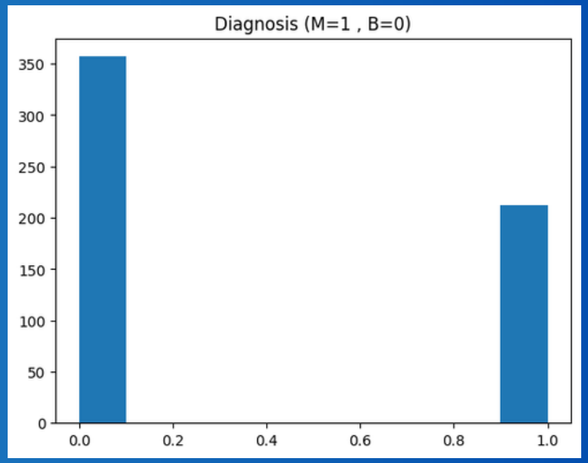
Modelo	Acurácia padrão	Acurácia melhorada
kNN	96%	98%
AS	94%	96,5%
RF	94%	95%

Conclusões

- **Pouca diferença entre as métricas dos modelos**
- **kNN um pouco melhor**
- **Taxas muito elevadas de acurácia!**

↓

- **Tamanho do dataset relativamente pequeno: 569 instâncias**
- Uso de validação cruzada para teste de diferentes subconjuntos de dados
- Matriz de confusão, recall e precisão também altos
- Procura por bons hiperparâmetros
- Dataset balanceado



Sistemas Inteligentes - IF684

Prof: Adriano Lorena

Obrigado pela atenção!

Projeto Aprendizado Supervisionado