**I. Pen-and-paper**

1. Cálculo dos *priors*

Função de densidade de probabilidade de condicionada às classes – Distribuição normal

Função de massa de probabilidade de condicionada às classes

Função de densidade de probabilidade de e condicionadas às classes – Distribuição normal multivariada

=

=

Utilizando a Regra de Bayes concluímos que:

Logo, utilizando o MAP obtemos que será classificado como 1 se:

1. Calculando as probabilidades condicionadas às classes dados determinados input obtemos,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | Classe Prevista | Classe Real |
|  |  |  |  | N | N |
|  |  |  |  | P | N |
|  |  |  |  | N | N |
|  |  |  |  | P | N |
|  |  |  |  | N | P |
|  |  |  |  | P | P |
|  |  |  |  | P | P |
|  |  |  |  | P | P |
|  |  |  |  | N | P |
|  |  |  |  | P | P |

Exemplificação dos cálculos:

será classificado como negativo pois a probabilidade obtida é inferior a 50%, sendo que na realidade pertence à classe dos negativos. Trata-se assim de um verdadeiro negativo.

Construção da matriz de confusão:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Valores Reais | |
| P | N |
| Valores Previstos | P | 4 | 2 |
| N | 2 | 2 |

1. O valor da medida F1 é dado por

Sendo que esta medida fica toma valores entre 0 e 1, sendo o valor 1 o caso ideal, podemos concluir que se trata de uma avaliação razoável do modelo treinado.

1. Organizando, por ordem crescente, os valores da probabilidade de as observações pertencerem à classe P, estabelecemos valores intermédios para utilizar como *treshold* e testar a eficácia do modelo em todos os valores possíveis. Os resultados obtidos encontram-se na seguinte tabela, onde as células sombreadas a verde significam uma previsão acertada, ao contrário das células sombreadas a verde.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *Tresholds* | | | | | | | | | | | Classe Real |
|  | 18 | 23 | 48 | 54 | 63 | 70 | 84 | 89 | 93 | 94 |
|  |  | P | N | N | N | N | N | N | N | N | N | N | N |
|  |  | P | P | N | N | N | N | N | N | N | N | N | P |
|  |  | P | P | P | N | N | N | N | N | N | N | N | N |
|  |  | P | P | P | P | N | N | N | N | N | N | N | P |
|  |  | P | P | P | P | P | N | N | N | N | N | N | P |
|  |  | P | P | P | P | P | P | N | N | N | N | N | N |
|  |  | P | P | P | P | P | P | P | N | N | N | N | N |
|  |  | P | P | P | P | P | P | P | P | N | N | N | P |
|  |  | P | P | P | P | P | P | P | P | P | N | N | P |
|  |  | P | P | P | P | P | P | P | P | P | P | N | P |
| Eficácia(%) | | 60 | 70 | 60 | 70 | 60 | 50 | 60 | 70 | 60 | 50 | 40 |  |

Assim podemos concluir que a eficácia é maximizada em

onde toma o valor de 70%

Assim, o *treshold* normalmente utilizado de 50% não se encontra dentro dos intervalos de maximização pelo que, neste caso não deve ser utilizado para obter os melhores resultados possíveis.

**II. Programming and critical analysis**



1. Utilizando o método de validação pretendido obtemos, aproximadamente, 96.63% de eficácia para k = 3, 97.36% de eficácia para k = 5 e 96.77% de eficácia para k = 7.

Após comparar a eficácia dos modelos ao serem testados com os dados com que foram treinados e com a fração dos dados não utilizada no treino, podemos observar que a eficácia destes é maior quando testados nos valores utilizados para treino. Neste caso, verificou-se, aproximadamente, 98.03% de eficácia para k = 3, 98.04% de eficácia para k = 5 e 97.93% de eficácia para k = 7.

Assim, podemos verificar que a diferença de eficácia é menor para k=5 o que nos leva à conclusão de que é o valor de k menos suscetível ao risco do *overfitting*.

1. Para testar a hipótese pretendida realizamos um *paired sampled t-test* unidirecional em que tentamos rejeitar a hipótese nula, “kNN não é estatisticamente superior ao Naïve Bayes”. Utilizando o método *10-fold cross validation* obtivemos um vetor com as eficácias de ambos os modelos ao serem aplicados sobre os mesmos dados. Realizando o teste, obtém-se um valor-p de 0.00015353096761064943 que nos permite, a qualquer nível de significância inferior a 0.015 %, rejeitar a hipótese nula, ou seja, afirmar que “kNN é estatisticamente superior ao Naïve Bayes”.

1. Answer 8

**III. APPENDIX**

**import** pandas **as** pd

**import** numpy **as** np

**from** scipy.io **import** arff

**from** sklearn.naive\_bayes **import** MultinomialNB

**from** sklearn **import** model\_selection, neighbors

**from** sklearn.metrics **import** accuracy\_score

**from** scipy **import** stats

data = arff.loadarff('breast.w.arff')

df = pd.DataFrame(data[0])

df.dropna(inplace=True)

df.replace(b'benign', 0, inplace=True)

df.replace(b'malignant', 1, inplace=True)

data = df.drop(["Class"],axis=1).values

target = df["Class"].values

fold = model\_selection.KFold(n\_splits=10, shuffle=True, random\_state=9)

NB = MultinomialNB()

scores\_KNN = [[],[],[]]

scores\_overfit\_KNN = [[],[],[]]

scores\_NB = []

i = 0

**for** k **in** [3,5,7]:

KNN=neighbors.KNeighborsClassifier(n\_neighbors=k,p=2,weights="uniform")

**for** train\_filter, test\_filter **in** fold.split(data):

data\_train, data\_test, target\_train, target\_test = data[train\_filter], data[test\_filter], target[train\_filter], target[test\_filter]

KNN.fit(data\_train,target\_train)

Predictions = KNN.predict(data\_test)

scores\_KNN[i].append(accuracy\_score(target\_test,predictions))

KNN.fit(data\_train,target\_train)

Predictions = KNN.predict(data\_train)

scores\_overfit\_KNN[i].append(accuracy\_score(target\_train,predictions))

**if** (k == 3):

NB.fit(data\_train,target\_train)

predictions=NB.predict(data\_test)

scores\_NB.append(accuracy\_score(target\_test,predictions))

i += 1

**print**("KNN accuracy means:")

**for** i **in** **range**(3):

**print**(" k =", 2\*i+3, np.array(scores\_KNN[i]).mean())

**print**()

overfit\_delta = []

**for** i **in** **range**(3):

overfit\_delta.append(np.array(scores\_overfit\_KNN[i]).mean() - np.array(scores\_KNN[i]).mean())

**print**("Less susceptible to overfitting: k =", 2\*overfit\_delta.index(**min**(overfit\_delta))+3)

**print**()

**print**("P-value:", stats.ttest\_rel(**list**(scores\_KNN[0]), scores\_NB, alternative="greater").pvalue)

**END**