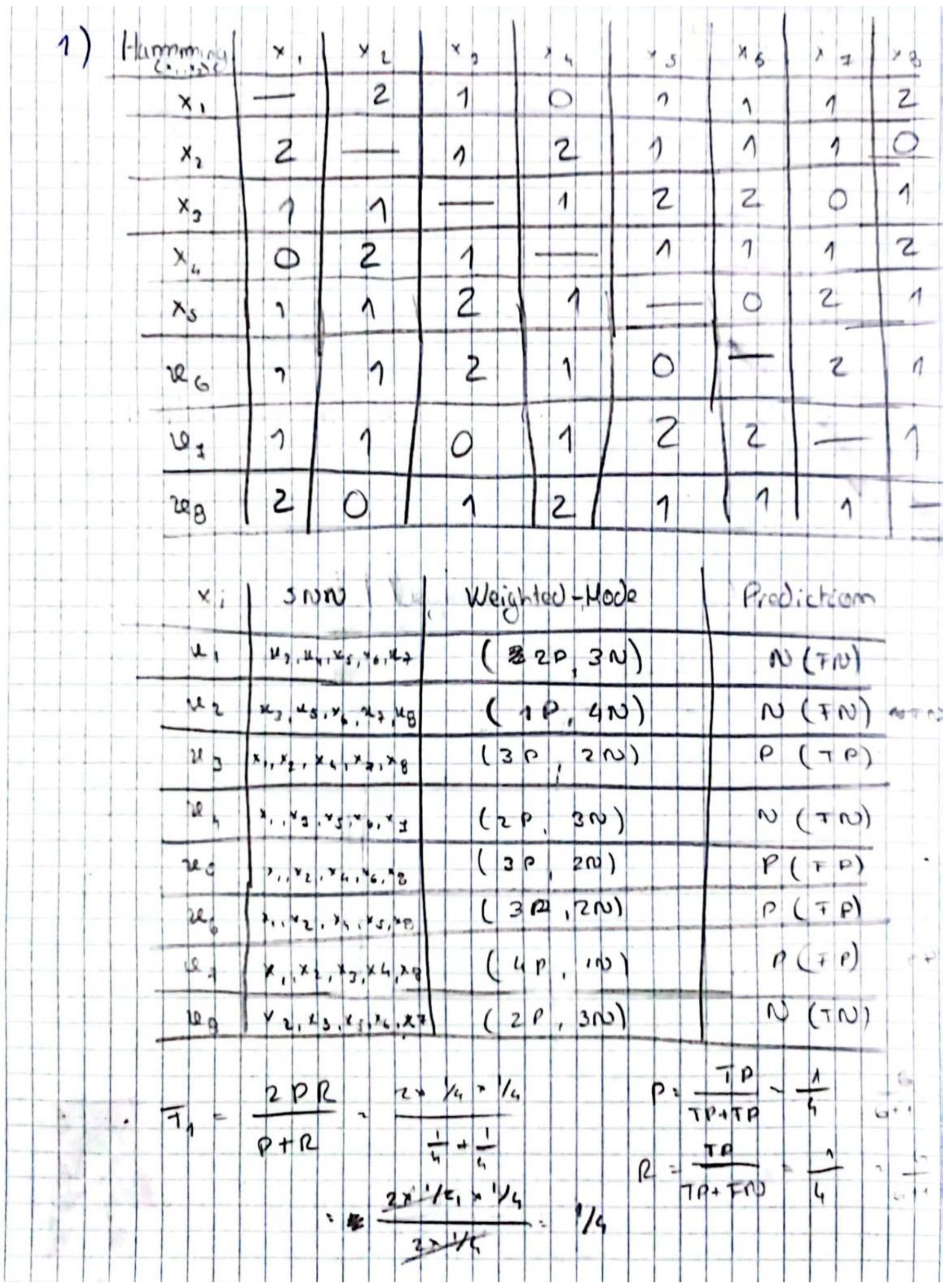
Aprendizagem 2022/23

Homework II – Group 066

(ist1106794, ist1107301)

I. Pen-and-paper

1)



1)

Hamming distance	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	—	2	1	0	1	1	1	2
x_2	2	—	1	2	1	1	1	0
x_3	1	1	—	1	2	2	0	1
x_4	0	2	1	—	1	1	1	2
x_5	1	1	2	1	—	0	2	1
x_6	1	1	2	1	0	—	2	1
x_7	1	1	0	1	2	2	—	1
x_8	2	0	1	2	1	1	1	—

x_i	5NN	Weighted-Mode	Prediction
x_1	x_3, x_4, x_5, x_6, x_7	(~~3~~ 2P, 3N)	N (FN)
x_2	x_3, x_5, x_6, x_7, x_8	(1P, 4N)	N (FN)
x_3	x_1, x_2, x_4, x_7, x_8	(3P, 2N)	P (TP)
x_4	x_1, x_3, x_5, x_6, x_7	(2P, 3N)	N (TN)
x_5	x_1, x_2, x_4, x_6, x_8	(3P, 2N)	P (FP)
x_6	x_1, x_2, x_4, x_5, x_8	(3P, 2N)	P (FP)
x_7	x_1, x_2, x_3, x_4, x_8	(4P, 1N)	P (FP)
x_8	x_2, x_3, x_5, x_6, x_7	(2P, 3N)	N (TN)

$$F_1 = \frac{2PR}{P+R} = \frac{2 \times 1/4 \times 1/4}{\frac{1}{4}+\frac{1}{4}} = \frac{\cancel{2 \times 1/4} \times 1/4}{\cancel{2 \times 1/4}} = 1/4$$

$$P = \frac{TP}{TP+TP} = \frac{1}{4}$$

$$R = \frac{TP}{TP+FN} = \frac{1}{4}$$

Answer: $\mathrm{F1} = \frac{1}{4}$

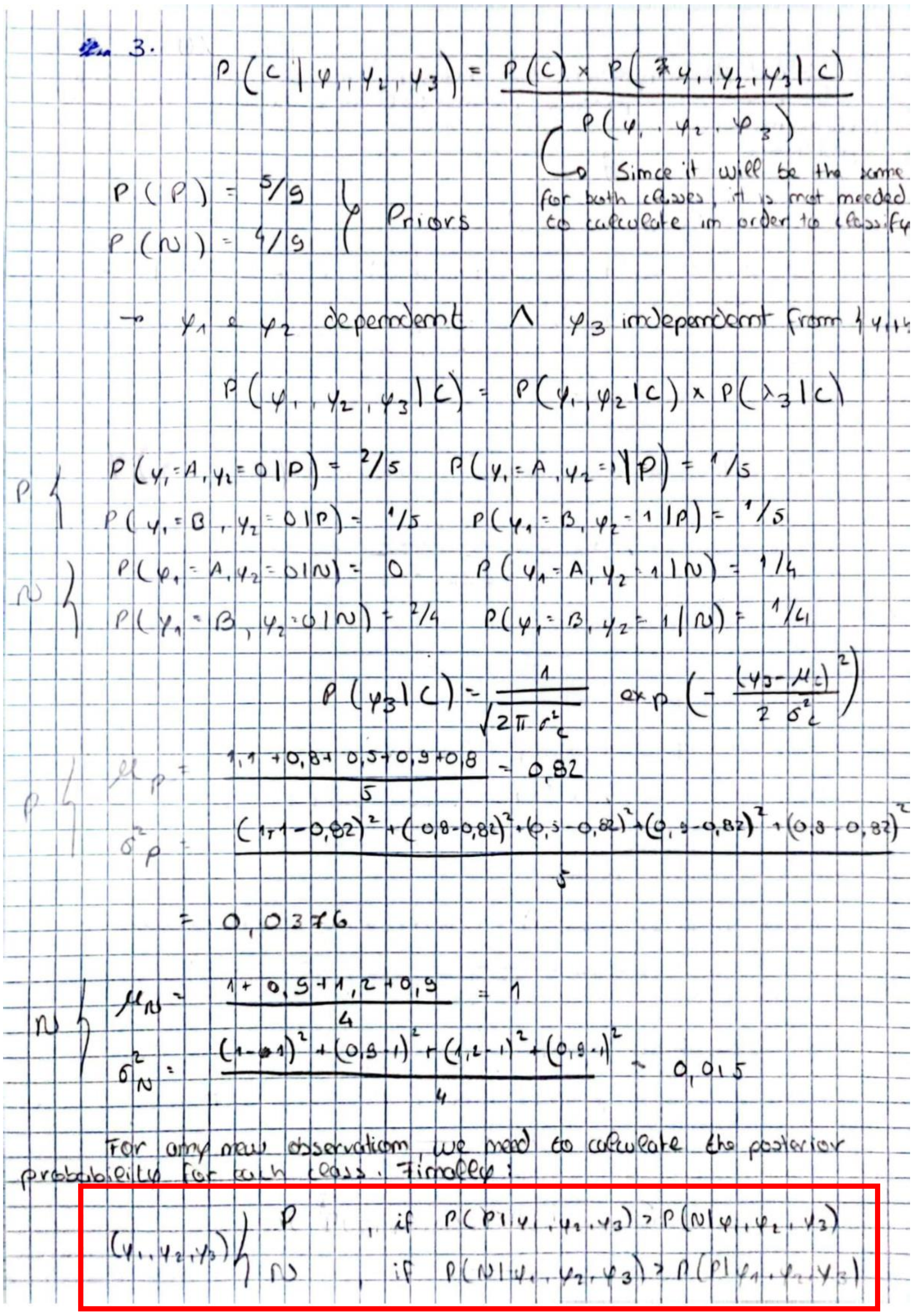
Aprendizagem 2022/23

Homework II – Group 066

(ist1106794, ist1107301)

2) We propose a custom weighted distance that emphasizes more on the categorical feature than the numeric one. As such, we suggest calculating the distance as $3 * (f_{i1} - f_{j1}) + |f_{i2} - f_{j2}|$, with f_1 representing the categorical feature and f_2 the numeric one. We then calculated the F1-measure using the new distance and obtained a score of 0,86 . It represents a 3,43 times better result than the previous 0.25 F1 score.

3)



3.

$$P(c \mid y_1, y_2, y_3) = \frac{P(c) \times P(y_1, y_2, y_3 \mid c)}{P(y_1, y_2, y_3)}$$

→ Since it will be the same for both classes, it is not needed to calculate in order to classify

$P(P) = 5/9$, $P(N) = 4/9$ } Priors

→ y_1 & y_2 dependent $\wedge$ y_3 independent from $\{y_1, y_2\}$

$$P(y_1, y_2, y_3 \mid c) = P(y_1, y_2 \mid c) \times P(y_3 \mid c)$$

P { $P(y_1 = A, y_2 = 0 \mid P) = 2/5$ $\quad$ $P(y_1 = A, y_2 = 1 \mid P) = 1/5$

$P(y_1 = B, y_2 = 0 \mid P) = 1/5$ $\quad$ $P(y_1 = B, y_2 = 1 \mid P) = 1/5$

N { $P(y_1 = A, y_2 = 0 \mid N) = 0$ $\quad$ $P(y_1 = A, y_2 = 1 \mid N) = 1/4$

$P(y_1 = B, y_2 = 0 \mid N) = 2/4$ $\quad$ $P(y_1 = B, y_2 = 1 \mid N) = 1/4$

$$P(y_3 \mid c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left(-\frac{(y_3 - \mu_c)^2}{2\sigma_c^2}\right)$$

P { $\mu_P = \frac{1,1 + 0,8 + 0,5 + 0,9 + 0,8}{5} = 0,82$

$\sigma_P^2 = \frac{(1,1 - 0,82)^2 + (0,8 - 0,82)^2 + (0,5 - 0,82)^2 + (0,9 - 0,82)^2 + (0,8 - 0,82)^2}{5}$

$= 0,0376$

N { $\mu_N = \frac{1 + 0,5 + 1,2 + 0,9}{4} = 1$

$\sigma_N^2 = \frac{(1-1)^2 + (0,5-1)^2 + (1,2-1)^2 + (0,9-1)^2}{4} = 0,015$

For any new observation, we need to calculate the posterior probability for each class. Finally:

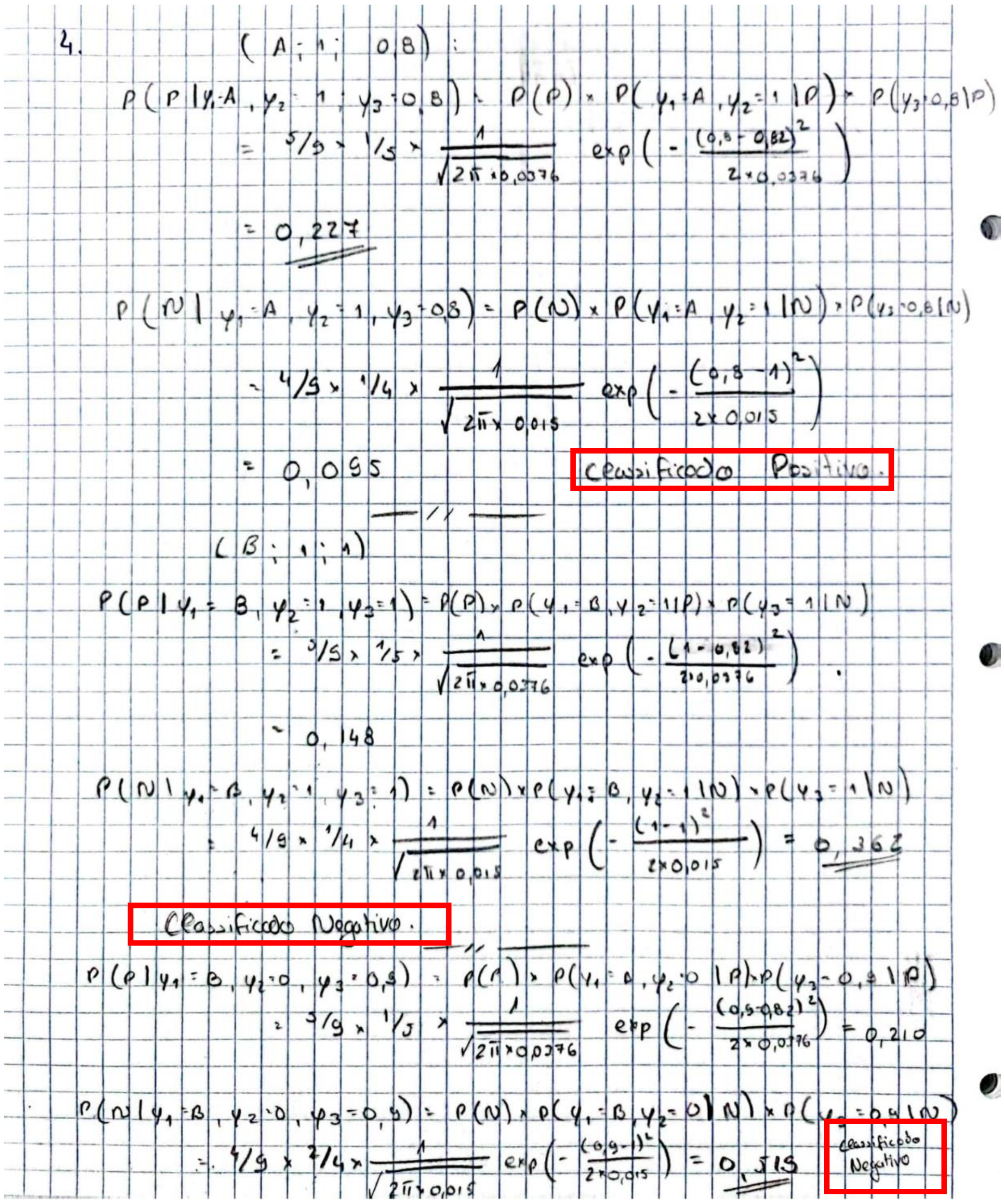
$$(y_1, y_2, y_3) \begin{cases} P & , \text{if } P(P \mid y_1, y_2, y_3) > P(N \mid y_1, y_2, y_3) \\ N & , \text{if } P(N \mid y_1, y_2, y_3) > P(P \mid y_1, y_2, y_3) \end{cases}$$

Aprendizagem 2022/23

Homework II – Group 066

(ist1106794, ist1107301)

4)



4. $(A; 1; 0{,}8)$:

$P(P \mid y_1 = A, y_2 = 1, y_3 = 0{,}8) = P(P) \times P(y_1 = A, y_2 = 1 \mid P) \times P(y_3 = 0{,}8 \mid P)$

$= 5/9 \times 1/5 \times \frac{1}{\sqrt{2\pi \times 0{,}0376}} \exp\left(-\frac{(0{,}8-0{,}82)^2}{2 \times 0{,}0376}\right)$

$= \underline{\underline{0{,}227}}$

$P(N \mid y_1 = A, y_2 = 1, y_3 = 0{,}8) = P(N) \times P(y_1 = A, y_2 = 1 \mid N) \times P(y_3 = 0{,}8 \mid N)$

$= 4/9 \times 1/4 \times \frac{1}{\sqrt{2\pi \times 0{,}015}} \exp\left(-\frac{(0{,}8-1)^2}{2 \times 0{,}015}\right)$

$= 0{,}095$ Classificado Positivo.

— // —

$(B; 1; 1)$

$P(P \mid y_1 = B, y_2 = 1, y_3 = 1) = P(P) \times P(y_1 = B, y_2 = 1 \mid P) \times P(y_3 = 1 \mid N)$

$= 5/9 \times 1/5 \times \frac{1}{\sqrt{2\pi \times 0{,}0376}} \exp\left(-\frac{(1-0{,}82)^2}{2 \times 0{,}0376}\right)$

$= 0{,}148$

$P(N \mid y_1 = B, y_2 = 1, y_3 = 1) = P(N) \times P(y_1 = B, y_2 = 1 \mid N) \times P(y_3 = 1 \mid N)$

$= 4/9 \times 1/4 \times \frac{1}{\sqrt{2\pi \times 0{,}015}} \exp\left(-\frac{(1-1)^2}{2 \times 0{,}015}\right) = \underline{\underline{0{,}362}}$

Classificado Negativo.

— // —

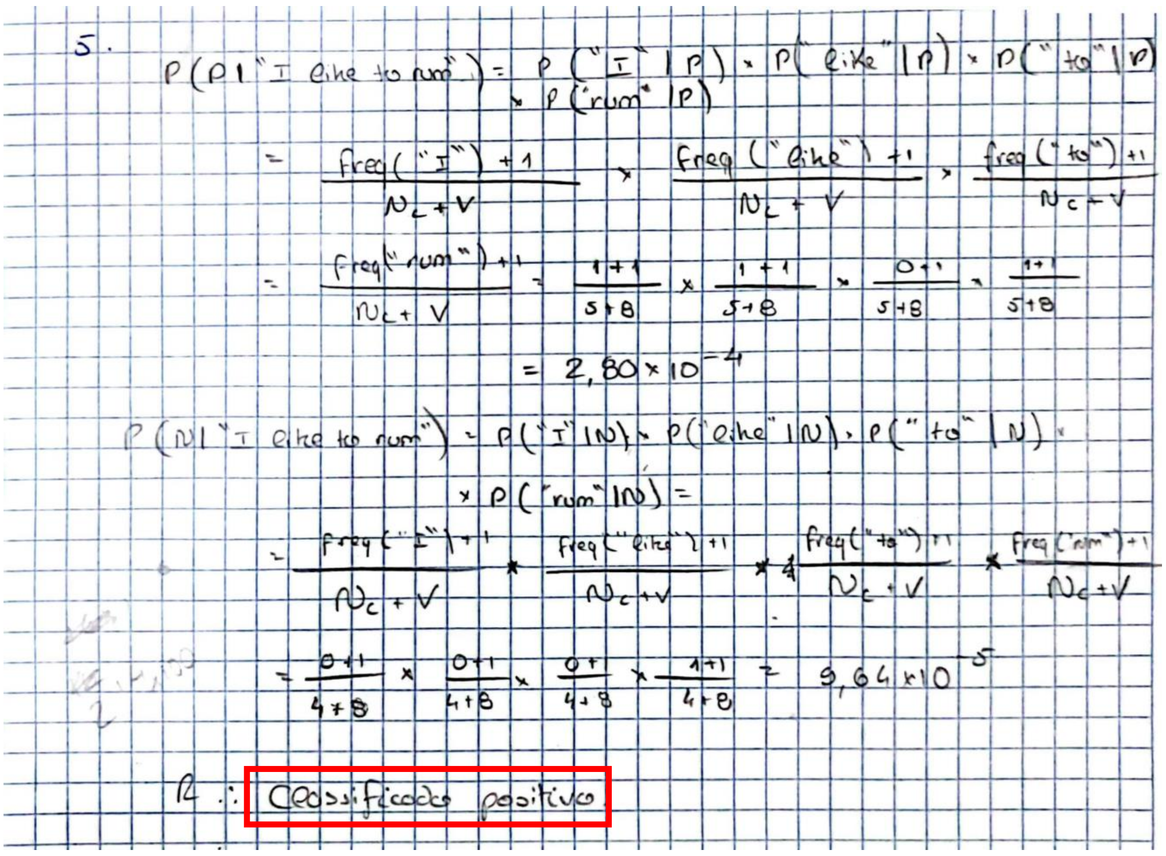
$P(P \mid y_1 = B, y_2 = 0, y_3 = 0{,}9) = P(P) \times P(y_1 = B, y_2 = 0 \mid P) \times P(y_3 = 0{,}9 \mid P)$

$= 5/9 \times 1/5 \times \frac{1}{\sqrt{2\pi \times 0{,}0376}} \exp\left(-\frac{(0{,}9-0{,}82)^2}{2 \times 0{,}0376}\right) = 0{,}210$

$P(N \mid y_1 = B, y_2 = 0, y_3 = 0{,}9) = P(N) \times P(y_1 = B, y_2 = 0 \mid N) \times P(y_3 = 0{,}9 \mid N)$

$= 4/9 \times 2/4 \times \frac{1}{\sqrt{2\pi \times 0{,}015}} \exp\left(-\frac{(0{,}9-1)^2}{2 \times 0{,}015}\right) = \underline{\underline{0{,}515}}$ Classificado Negativo

5)



5.

$$P(P \mid \text{"I like to run"}) = P(\text{"I"} \mid P) \times P(\text{"like"} \mid P) \times P(\text{"to"} \mid P) \times P(\text{"run"} \mid P)$$

$$= \frac{freq(\text{"I"})+1}{N_c+V} \times \frac{freq(\text{"like"})+1}{N_c+V} \times \frac{freq(\text{"to"})+1}{N_c+V} \times \frac{freq(\text{"run"})+1}{N_c+V} = \frac{1+1}{5+8} \times \frac{1+1}{5+8} \times \frac{0+1}{5+8} \times \frac{1+1}{5+8}$$

$$= 2,80 \times 10^{-4}$$

$$P(N \mid \text{"I like to run"}) = P(\text{"I"} \mid N) \cdot P(\text{"like"} \mid N) \cdot P(\text{"to"} \mid N) \times P(\text{"run"} \mid N) =$$

$$= \frac{freq(\text{"I"})+1}{N_c+V} \times \frac{freq(\text{"like"})+1}{N_c+V} \times \frac{freq(\text{"to"})+1}{N_c+V} \times \frac{freq(\text{"run"})+1}{N_c+V}$$

$$= \frac{0+1}{4+8} \times \frac{0+1}{4+8} \times \frac{0+1}{4+8} \times \frac{1+1}{4+8} = 9,64 \times 10^{-5}$$

R.: $\boxed{\text{Classificado positivo}}$

Aprendizagem 2022/23

Homework II – Group 066

(ist1106794, ist1107301)

II. Programming and critical analysis

Here we have an overview of all the imports used and how we imported the dataset, which are both common to all questions. Therefore, we will be omitting this code from the beginning of each task to avoid redundancy.

```python
import pandas as pd, matplotlib.pyplot as plt, numpy as np
import seaborn as sns
import scipy.stats as stats
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the data
df = pd.read_csv('heart-disease.csv')
X = df.drop('target', axis=1)
y = df['target']

folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
```

1)

a)

Classifying the dataset using both approaches

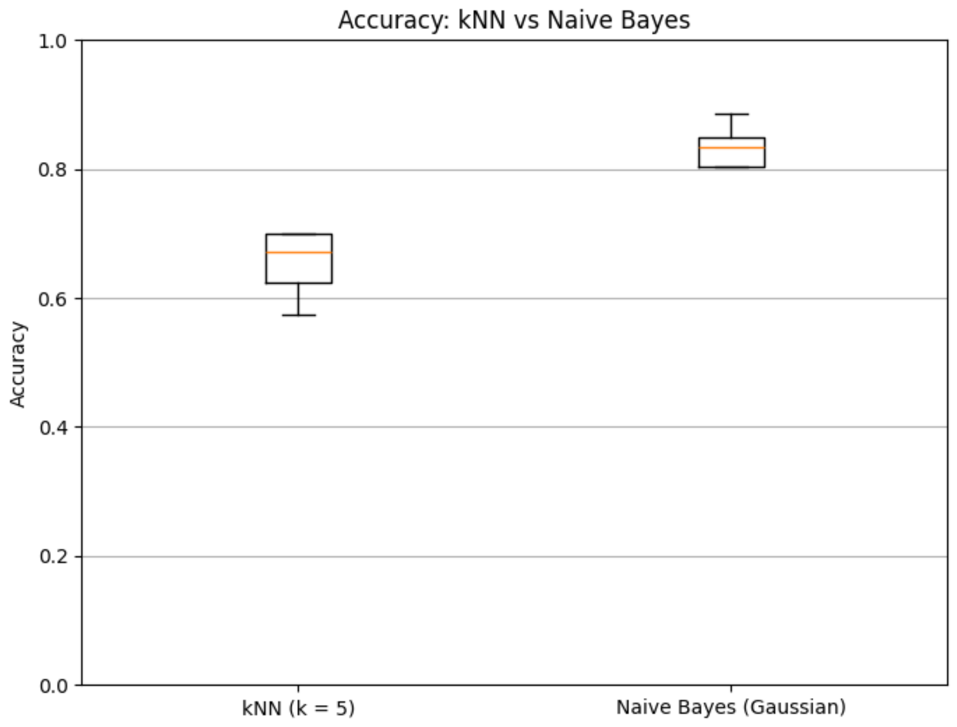
```python
## Classify the data set using kNN and Naive Bayes
knn = KNeighborsClassifier(n_neighbors=5)
nb = GaussianNB()

## Evaluate by cross-validation
knn_scores = cross_val_score(knn, X, y, cv=folds)
nb_scores = cross_val_score(nb, X, y, cv=folds)

print(f"Scaled kNN Accuracy: {knn_scores.mean():.3f}")
print(f"Scaled Naive Bayes Accuracy: {nb_scores.mean():.3f}")
```

Printing the boxplots

```
# Plot the figure
plt.figure(figsize=(8, 6))
plt.boxplot([knn_scores, nb_scores], tick_labels=['kNN (k = 5)', 'Naive Bayes (Gaussian)'] )
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.title('Accuracy: kNN vs Naive Bayes')
plt.grid(axis = 'y')
plt.show()
```

Output:

```
Scaled kNN Accuracy: 0.654
Scaled Naive Bayes Accuracy: 0.835
```

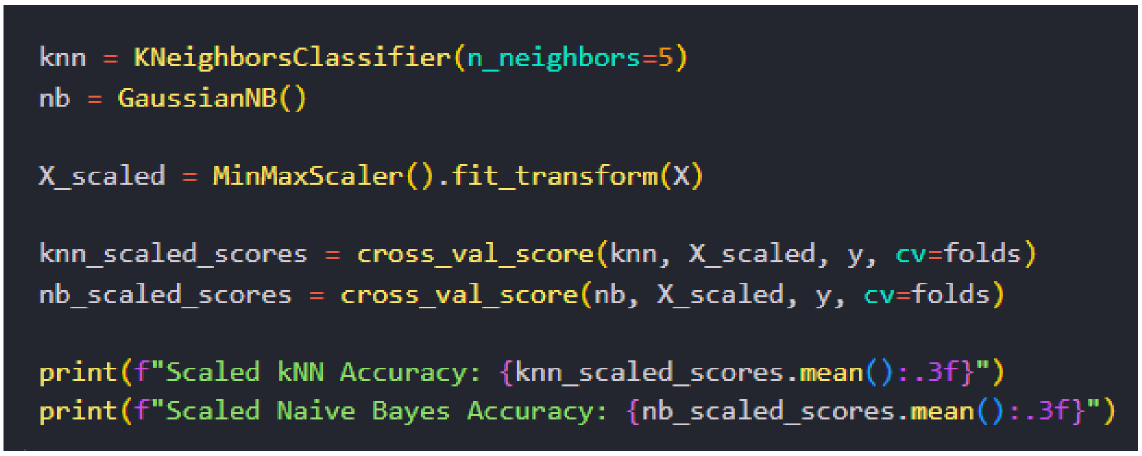


The Naïve Bayes Gaussian model appears to be more stable when it comes to accuracy compared to the kNN model. This might be because of the high dimensionality of the data set (13 features). The kNN model is distance-based, which means it relies on distance between data points to identify the relationship between them and patterns in the data. For high-dimensional spaces, the data becomes too sparse and the distance between instances more insignificant, as they all become very far from one another. Therefore, the kNN model struggles to find "closeness" between data instances and suffers in accuracy scores.

On the other hand, by assuming independence between features, the Naïve Bayes model becomes simpler and may generalize better, leading to similar accuracies across all folds.

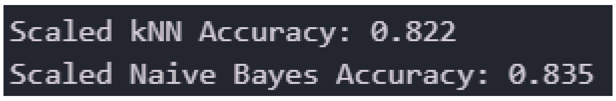
b)

Classifying and scaling the data



```
knn = KNeighborsClassifier(n_neighbors=5)
nb = GaussianNB()

X_scaled = MinMaxScaler().fit_transform(X)

knn_scaled_scores = cross_val_score(knn, X_scaled, y, cv=folds)
nb_scaled_scores = cross_val_score(nb, X_scaled, y, cv=folds)

print(f"Scaled kNN Accuracy: {knn_scaled_scores.mean():.3f}")
print(f"Scaled Naive Bayes Accuracy: {nb_scaled_scores.mean():.3f}")
```

Output:



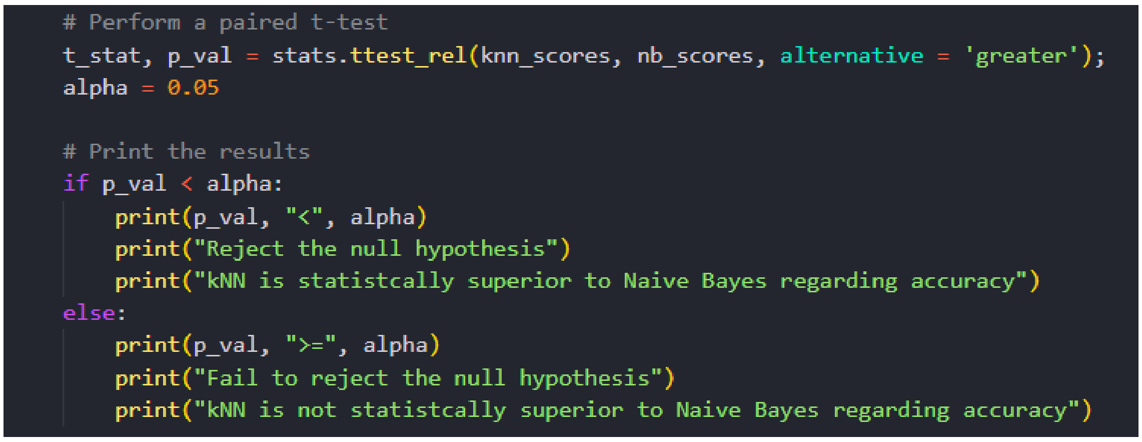
```
Scaled kNN Accuracy: 0.822
Scaled Naive Bayes Accuracy: 0.835
```

Since the features in the data set have different scales, they will end up impacting the results in the kNN model with different significance, due to it being a distance-based model. This introduces a bias in the data set. However, the Naïve Bayes model is not affected by this problem because it assumes independence between the features.

By applying Min-Max Scaling during the preprocessing stage, we aim to reduce the bias in the kNN algorithm by normalizing all features under the same scale (usually [0, 1]). Now, they should all have the same impact in the distance calculation.

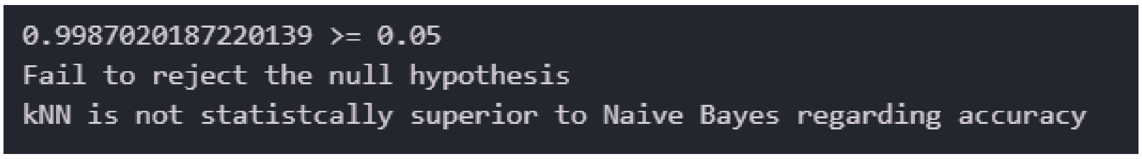
The impact in the accuracy after applying Min-Max Scaling proves the theory, as we can see that the accuracy for the kNN model increased significantly and the accuracy for the Naïve Bayes model stayed pretty much the same.

c) Performing the t-test



```
# Perform a paired t-test
t_stat, p_val = stats.ttest_rel(knn_scores, nb_scores, alternative = 'greater');
alpha = 0.05

# Print the results
if p_val < alpha:
    print(p_val, "<", alpha)
    print("Reject the null hypothesis")
    print("kNN is statistcally superior to Naive Bayes regarding accuracy")
else:
    print(p_val, ">=", alpha)
    print("Fail to reject the null hypothesis")
    print("kNN is not statistcally superior to Naive Bayes regarding accuracy")
```

Output:

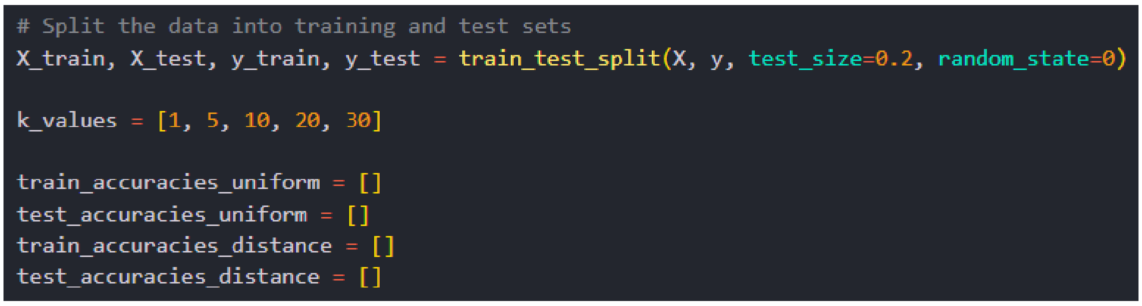


```
0.9987020187220139 >= 0.05
Fail to reject the null hypothesis
kNN is not statistcally superior to Naive Bayes regarding accuracy
```

2)

a)

Splitting the data into and initializing all necessary variables



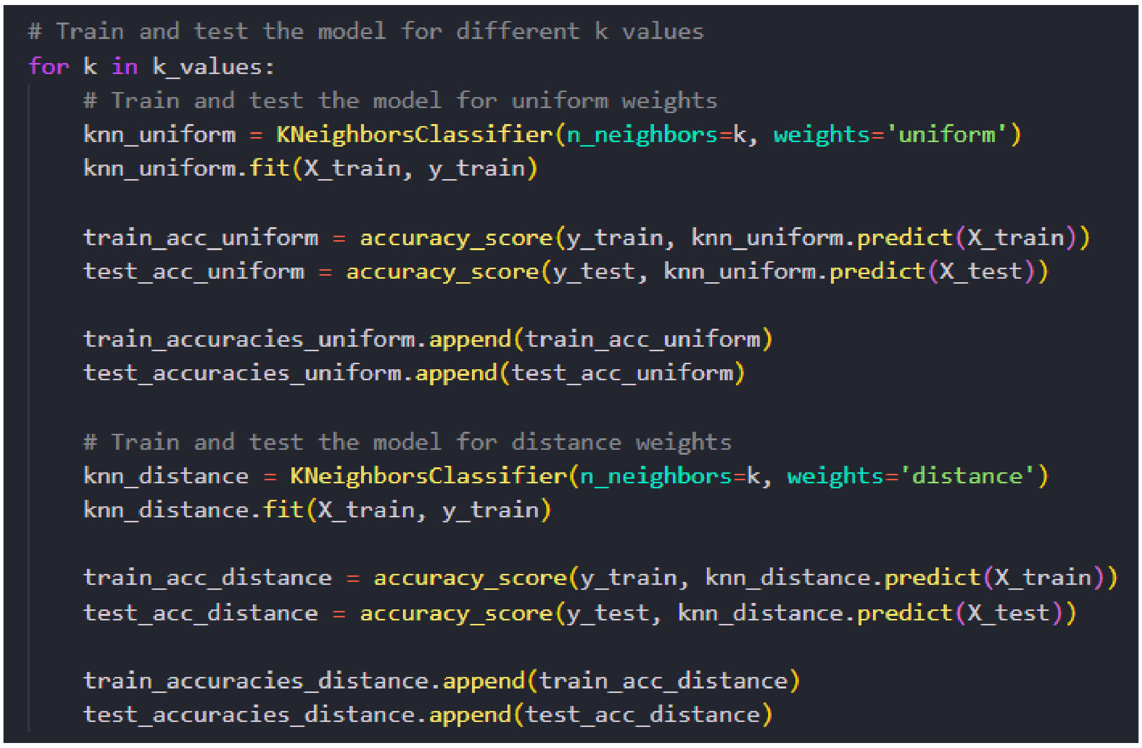
```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

k_values = [1, 5, 10, 20, 30]

train_accuracies_uniform = []
test_accuracies_uniform = []
train_accuracies_distance = []
test_accuracies_distance = []
```

Aprendizagem 2022/23

Homework II – Group 066

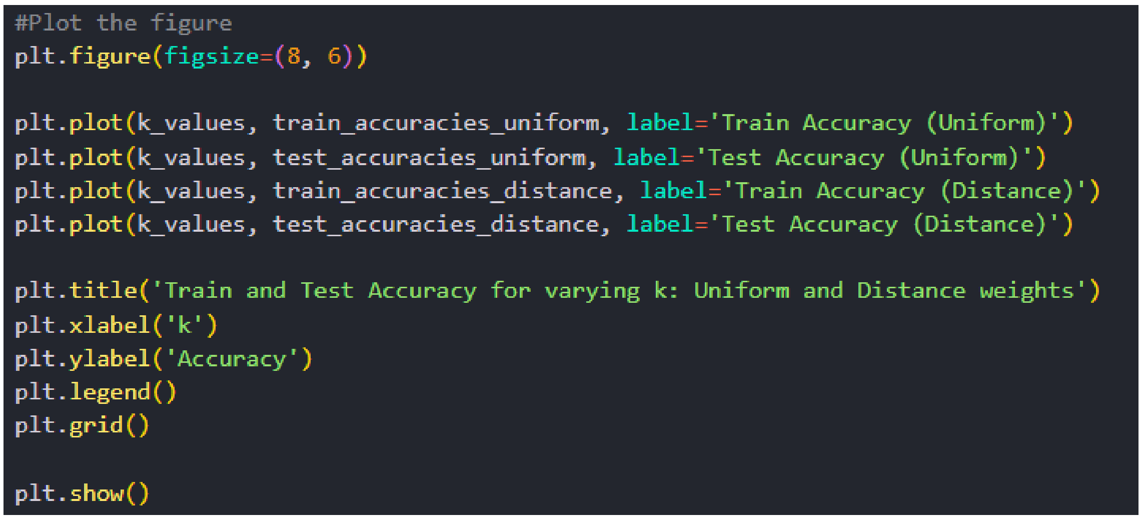
(ist1106794, ist1107301)

Training, testing and evaluating the accuracy of the model for all k values



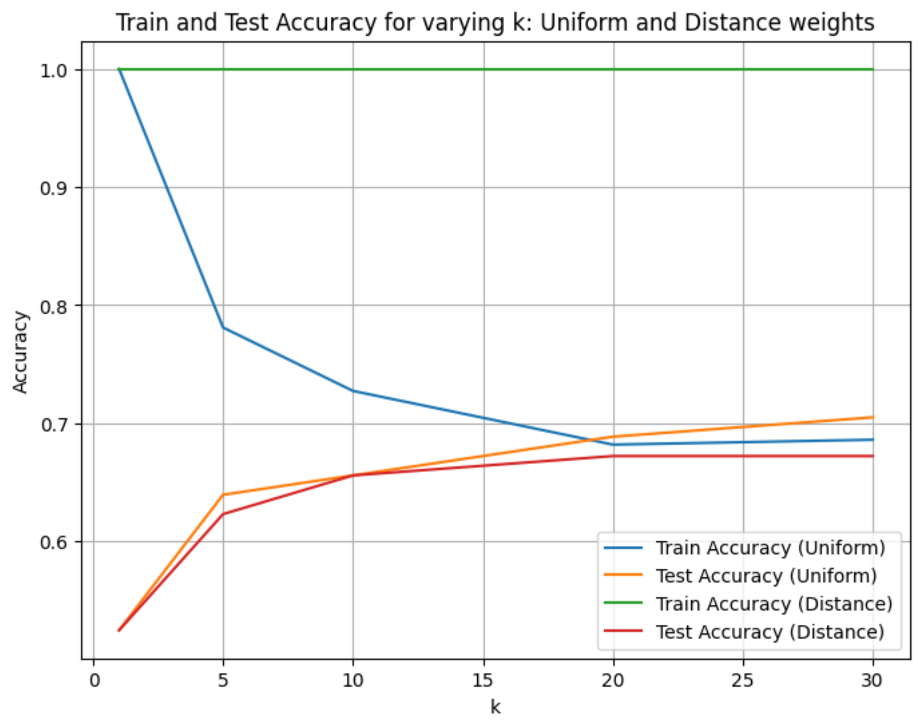
```python
# Train and test the model for different k values
for k in k_values:
    # Train and test the model for uniform weights
    knn_uniform = KNeighborsClassifier(n_neighbors=k, weights='uniform')
    knn_uniform.fit(X_train, y_train)

    train_acc_uniform = accuracy_score(y_train, knn_uniform.predict(X_train))
    test_acc_uniform = accuracy_score(y_test, knn_uniform.predict(X_test))

    train_accuracies_uniform.append(train_acc_uniform)
    test_accuracies_uniform.append(test_acc_uniform)

    # Train and test the model for distance weights
    knn_distance = KNeighborsClassifier(n_neighbors=k, weights='distance')
    knn_distance.fit(X_train, y_train)

    train_acc_distance = accuracy_score(y_train, knn_distance.predict(X_train))
    test_acc_distance = accuracy_score(y_test, knn_distance.predict(X_test))

    train_accuracies_distance.append(train_acc_distance)
    test_accuracies_distance.append(test_acc_distance)
```

Printing the plot



```python
#Plot the figure
plt.figure(figsize=(8, 6))

plt.plot(k_values, train_accuracies_uniform, label='Train Accuracy (Uniform)')
plt.plot(k_values, test_accuracies_uniform, label='Test Accuracy (Uniform)')
plt.plot(k_values, train_accuracies_distance, label='Train Accuracy (Distance)')
plt.plot(k_values, test_accuracies_distance, label='Test Accuracy (Distance)')

plt.title('Train and Test Accuracy for varying k: Uniform and Distance weights')
plt.xlabel('k')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()

plt.show()
```

Output:



b) By increasing the number of neighbors in aa kNN model, it becomes a better at generalization. This is true because we look at a bigger portion of the population when deciding the output value of a point, instead of just the few closest points. Therefore, the model gets a better view of the data and patterns instead of focusing just on smaller areas and most probably catching noise during training.

The results confirm this hypothesis. We can see that for smaller values of k in the graph, the accuracy of the uniform weights training is very high, opposed to the small scores in the corresponding uniform weights test. The model is suffering from overfitting caused by the noise caught associated with only looking at the few closest neighbors. As we increase the k value, the test accuracy increases proving the also increasing generalization capability of the model.

3) First, when we use Naïve Bayes, we are assuming that all variables are independent from each other , which may not be true for our specific dataset. For example, features such as "cholesterol in mg/d", "resting blood pressure" and "age" are often not independent, as age is usually correlated with both these features. Ignoring this correlation may lead to misjudges by our model. For instance, high cholesterol and high blood pressure together could be stronger indicators of heart disease than when analyzed independently, something that Naïve Bayes wouldn't be able to catch.

Secondly, with Naïve Bayes we assume that all features are equally important, which may not reflect reality. Certain features, such as "chest pain" or "maximum heart rate achieved" may hold more predictive weight than "resting electrocardiographic results", when it comes to diagnose a heart disease.

END