CS464 Introduction to Machine Learning

Fall 2022

Homework 1

Emre Can Şen-21902516

P(H) = 64%, P($S_M \cap H$) = 64%*87%= 55.68%, P($S_U \cap$H) = 64%*(100-87=13)% = 8.32%

P(L) = 24%, P($S_M \cap L$) = 24%*21%= 5.04%, P($S_U \cap$L) = 24%*79%= 18.96%

P(F) = 12%, P($S_M \cap$F) = 12%*4%= 0.48%, P($S_U \cap$F) = 12%*96%= 11.52%

P($S_M|H$)= 87%, P($S_U|H$)= (100-87)=13%,

P($S_M|L$) = 21%, P($S_U|L$)= 79%

P($S_M|F$) = 4%, P($S_U|F$)= 96%

Q-1.1

P($S_M$) = P($S_M \cap H$)+ P($S_M \cap L$)+ P($S_M \cap$F)= 61.2%, P($S_U$)= (100-61.2=38.8)%

Q-1.2

$$\mathrm{P}(\mathrm{H}|S_M) = \frac{\mathrm{P}(S_M|H) * \mathrm{P}(\mathrm{H})}{\mathrm{P}(S_M)}$$

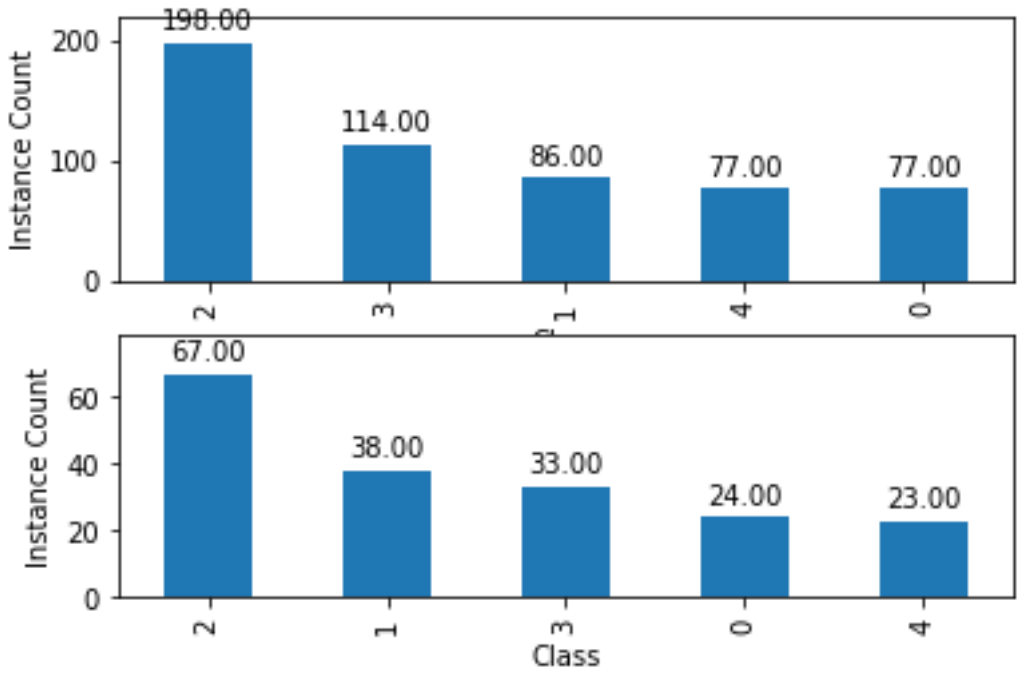
$$\mathrm{P}(\mathrm{H}|S_M) = \frac{\mathrm{P}(S_M|H) * \mathrm{P}(\mathrm{H})}{\mathrm{P}(S_M \cap H) + \mathrm{P}(S_M \cap L) + \mathrm{P}(S_M \cap \mathrm{F})} = \frac{87\% * 64\%}{61.2\%} = 90.9804\%$$

Q-1.3

$$\mathrm{P}(\mathrm{H}|S_U) = \frac{\mathrm{P}(S_U|H) * \mathrm{P}(\mathrm{H})}{\mathrm{P}(S_U \cap H) + \mathrm{P}(S_U \cap L) + \mathrm{P}(S_U \cap \mathrm{F})} = \frac{13\% * 64\%}{38.8\%} = 21.4433\%$$

Q.2.1

1.




2. It is skewed towards the class 2. Then class 3, then the class 1, and class 0 and 4 have the same weight. It would be a problem if the given data is not enough to eliminate the luck factor. If there is sufficient data, meaning there is no lucky set, then it being skewed is not such a problem.

3. Yes, they have similar distributions as it can be seen from the graph in the first part. The term $P(y = y_k)$ is misleading. The probability of classes would vary between populations and it would lead to confusion.

4. If the dataset for the lower sampled classes is enough, there shouldn't be noticeable accuracy problems. Otherwise low sample classes might have low accuracy.

Q2.2

Accuracy: 31.3514

Confusion Matrix:

[[24. 0. 0. 0. 0.]

[33. 5. 0. 0. 0.]

[45. 0. 22. 0. 0.]

[30. 0. 0. 3. 0.]

[19. 0. 0. 0. 4.]]

A lot of values are –inf. So automatically the code considers them equal value and picks the class 0. Hence there are a lot of class 0 predictions and most of them are wrong.

Q2.3

Accuracy: 97.2973

Confusion Matrix:

[[24.  0.  0.  0.  0.]

 [ 0. 35.  2.  1.  0.]

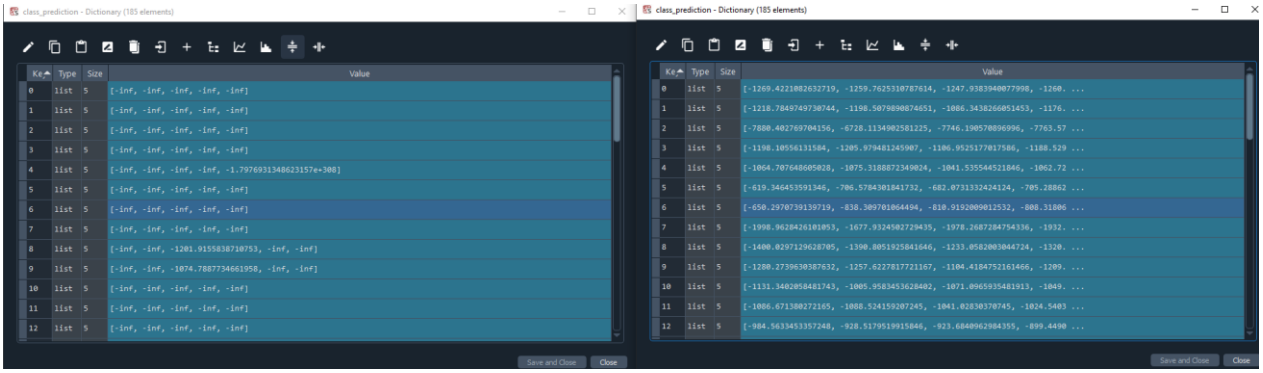
 [ 0.  0. 66.  1.  0.]

 [ 0.  0.  0. 33.  0.]

 [ 1.  0.  0.  0. 22.]]

Q2.4

Now that –inf values are replaced with small numbers, computer can work with this small number to compute the $y_i$ value and determine the greater value. Unlike the first case now all of the values do not have the same value –inf.




As it can be seen in the figure, -inf values are replaced with sufficiently small numbers and hence computer can distinguish the greater value. So greater accuracy is achieved.

Appendix

PART-1

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 26 17:23:46 2022

@author: Emre
"""
```

```
import matplotlib.pyplot as pp
import pandas as pd
import numpy as np

train_data = pd.read_csv("bbcsports_train.csv")
val_data = pd.read_csv("bbcsports_val.csv")
train_replaced = train_data.drop('class_label', inplace=False, axis=1)
val_replaced = val_data.drop('class_label', inplace=False, axis=1)

class_dict = pd.Series.to_dict(train_data.class_label.value_counts())

# plt.figure(1)
# plt.subplot(2,1,1)
# plot=train_data.class_label.value_counts().plot.bar(ylabel="Instance Count", xlabel="Class",
ylim=(0,220))

# for bar in plot.patches:
#     plot.annotate(format(bar.get_height(), '.2f'),
#                   (bar.get_x() + bar.get_width() / 2,
#                    bar.get_height()), ha='center', va='center',
#                   size=10, xytext=(0, 8),
#                   textcoords='offset points') [1]


# plt.subplot(2,1,2)

# plot2=val_data.class_label.value_counts().plot.bar(ylabel="Instance Count",
xlabel="Class",ylim=(0,79))

# for bar in plot2.patches:
```

```
#     plot2.annotate(format(bar.get_height(), '.2f'),
#                    (bar.get_x() + bar.get_width() / 2,
#                     bar.get_height()), ha='center', va='center',
#                    size=10, xytext=(0, 8),
#                    textcoords='offset points') [1]

article_length = len(train_data)
pi = {}
T_0 = {}

for x, values in class_dict.items():
    pi[x] = values/article_length

for classes in (0,1,2,3,4):
    class_data = train_data[train_data.class_label==classes]
    class_data = class_data.drop('class_label', inplace=False, axis=1)
    amount_list = []
    for column in class_data:
        amount_list.append(sum(class_data[column]))
    T_0[classes] = amount_list

T_val = pd.DataFrame.from_dict(T_0, orient='index')
Theta_value = {}
Series_sum = T_val.sum(axis=1)
Series_sum_list = pd.Series.to_list(Series_sum)
Theta_value = T_val.div(Series_sum, axis=0)

class_prediction = {}
for file in range(len(val_replaced)):
    prediction_values = []
```

```
    for clas in (0,1,2,3,4):
        a = np.nan_to_num(np.log(Theta_value.iloc[[clas]].values))
        y_value = np.log(pi[clas]) + np.sum(a*val_replaced.iloc[[file]].values)
        prediction_values.append(y_value)
    class_prediction[file] = prediction_values

predicted_classes = {}
for x in class_prediction:
    max_index = class_prediction[x].index(max(class_prediction[x]))
    predicted_classes[x] = max_index

true_classes = val_data.pop('class_label')
t = 0
f = 0
for x in predicted_classes:
    if predicted_classes[x] == true_classes.values[x]:
        t += 1
    else:
        f += 1

print("Accuracy Percentage:",t/(t+f)*100)

conf_vec=np.zeros((5,5))
for x in range(len(val_replaced)):
    conf_vec[true_classes[x]][predicted_classes[x]] +=1
print("Confusion Vector:\n",conf_vec)
```

PART-2

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 26 17:23:46 2022

@author: Emre
"""
import pandas as pd
import numpy as np

train_data = pd.read_csv("bbcsports_train.csv")
val_data = pd.read_csv("bbcsports_val.csv")
train_replaced = train_data.drop('class_label', inplace=False, axis=1)
val_replaced = val_data.drop('class_label', inplace=False, axis=1)

class_dict = pd.Series.to_dict(train_data.class_label.value_counts())

article_length = len(train_data)
pi = {}
T_0 = {}
alpha = 1

for x, values in class_dict.items():
    pi[x] = values/article_length

for classes in (0,1,2,3,4):
    class_data = train_data[train_data.class_label==classes]
    class_data = class_data.drop('class_label', inplace=False, axis=1)
    amount_list = []
    for column in class_data:
        amount_list.append(sum(class_data[column]))
```

```
    T_0[classes] = amount_list


T_val = pd.DataFrame.from_dict(T_0, orient='index')
T_val=T_val+alpha
Theta_value = {}
Series_sum = T_val.sum(axis=1)
Series_sum_list = pd.Series.to_list(Series_sum)
Theta_value = T_val.div(Series_sum+alpha*len(train_replaced.columns), axis=0)

class_prediction = {}
for file in range(len(val_replaced)):
    prediction_values = []
    for clas in (0,1,2,3,4):
        a = np.nan_to_num(np.log(Theta_value.iloc[[clas]].values))
        y_value = np.log(pi[clas]) + np.sum(a*val_replaced.iloc[[file]].values)
        prediction_values.append(y_value)
    class_prediction[file] = prediction_values

predicted_classes = {}
for x in class_prediction:
    max_index = class_prediction[x].index(max(class_prediction[x]))
    predicted_classes[x] = max_index

true_classes = val_data.pop('class_label')
t = 0
f = 0
for x in predicted_classes:
    if predicted_classes[x] == true_classes.values[x]:
        t += 1
    else:
```

```
        f += 1


print("Accuracy Percentage:",t/(t+f)*100)


conf_vec=np.zeros((5,5))

for x in range(len(val_replaced)):

    conf_vec[true_classes[x]][predicted_classes[x]] +=1

print("Confusion Vector:\n",conf_vec)
```

References


[1]*Pandas.dataframe.plot.barh#* (no date) *pandas.DataFrame.plot.barh - pandas 1.5.1 documentation*. Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.barh.html (Accessed: October 30, 2022).