Classification

In this exercise we are going through the application of different classification methods and related concepts.

Submission

In order to submit on gradescope, you need to submit the following:

- the homework jupyter notebook it self `hw7.ipynb`
- the pdf generated from the notebook, you can get the pdf from `File->Print Preview`
- the .py file generated from the notebook, you can get the .py file from `File->Download as->Python(.py)`

1. QMNIST Classification

In [ ]:
```python
# import some libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sn
import pickle
np.set_printoptions(suppress=True)
```

1.1 Load your data

In [ ]:
```python
...
train_data = ...
train_labels = ...

test_data = ...
test_labels = ...
```

In [ ]:
```python
train_data.shape, test_data.shape
```

1.2 Plot your data

Each QMNIST data point represents a 28 * 28 pixel hand written digit. Complete the following code to plot the first five data point from the train dataset.

In [ ]:
```python
# np.array.reshape might be useful.
fig, axes = plt.subplots(1, 5, figsize=(6, 6))
fig.tight_layout()

for i in range(5):
    ...
    axes[i].axis('off')

plt.show()
```

1.3 Naive Bayes

sklearn has two different implementions of naive bayes that we can use for this problem:

- `CategoricalNB()`
- `GaussianNB()`

Let's take a loot at both of them.

In [ ]:
```python
from sklearn.naive_bayes import CategoricalNB, GaussianNB
```

1.3.1 CategoricalNB

In CategoricalNB, we assume that each feature in the dataset is categorical. Therefore, the The probability of category t in feature i given class c is estimated as:

$$P(x_i = t|y = c; \alpha) = \frac{N_tic+\alpha}{N_c+\alpha n_i}$$

This is just what we see in class, with α being the smoothing variable.

Your task here is to draw out the test accuracy score for α between 0 and 2 with step to be 0.1.

In [ ]:
```python
# train models
...
```

In [ ]:
```python
# make the plot
...
```

Question: Describe how test accuracy changes and explain.

1.3.2 GaussianNB

In GaussainNB, we no longer assume each feature is categorical. Instead, we assume the likelyhood of each feature follows a Gaussain Distribution:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Again draw out the test accuracy score for $var\_smoothing$ between 0 and 2 with step to be 0.1.

In [ ]:
```python
# train models
```

In [ ]:
```python
# make the plot
...
```

Question: Compare the performance of the model with $var\_smoothing = 0$ and the model with $var\_smoothing$ being other values. What do you find? How would you explain this senario?

Hint: Take a look at the warning messages generated when you run the models. sklearn documentation might also be useful.

Question: Comparing the test accuracy between the above two different Naive Bayes models. Which one has a relatively low score? What might be the cause?

1.4 Confusion Matrix

In this section, we would like to analyze the confusion matrix of a given model.

The following is the definition of a confusion matrix:

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

For confusion matrix, use `sklearn.metrics.confusion_matrix`

Firstly, compute confusion matrix using `CategoricalNB` with $\alpha = 0.5$ and test data.

In [ ]:
```python
...
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix

labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
...
df_cm = pd.DataFrame(cm, index = [i for i in labels],
                     columns = [i for i in labels])
# print(df_cm)
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True)
```

Question: What are the top five confusing pairs (i -> j) of classes for you model? What might be the cause?

2. Binary Classification

Now we have gone over a problem with 10 classes. Let's take a deeper look at the classification on a relatively simple dataset.

2.1 Binary Dateset 1

Note that we have two binary datasets in the fold you downloaded. Let's first take a look at the first one.

In [ ]:
```python
df = pd.read_csv('binary_dataset1.csv')
df.head()
```

Let's take a look at the class distribution. Your task here is to draw a bar chart with each bar representing a simple class.

In [ ]:
```python
# Make the bar chart
...
```

Before training any model, you need to get the X and y out of the dataframe and do train test split

In [ ]:
```python
# Get X and y from the dataset
...
```

In [ ]:
```python
# Do train test split with random_state=0 and test_size=0.5
...
```

2.1.1 Naive Bayes in Binary Classification

First we want to solve this problem using Naive Bayes model with proper parameter. Choose a proper Naive Bayes class to solve this priblem. Report your test accuracy.

In [ ]:
```python
...
```

2.1.2 Using Regression for Classification

Since this is a binary classification problem, we can solve it in the following steps:

1. Fit a linear regression model
2. Get your raw predicted values x_r from the model
3. Find a threshold ϵ in a way such that
4. Get your final predicted class x_c in a way that $x_c = 0$ if $x_r < \epsilon$, otherwsie $x_c = 1$

One way to choose the ϵ here is to find the one that maximize the train accuracy, and then apply to test data.

So your task here is to create such a model, find ϵ, and report test accuracy.

In [ ]:
```python
from sklearn.linear_model import LinearRegression
```

In [ ]:
```python
...
```

Write down what you find here:

- ϵ =
- train_accuracy =
- test_accuracy =

Question: Can you think of any model you learnt from class that is similar to this way of doing classification?

2.2 Binary dataset 2

Then let's take a look at the second binary dataset.

As usual, we take a look at the class distribution.

In [ ]:
```python
df = pd.read_csv('binary_dataset2.csv')
df.head()
```

In [ ]:
```python
...
```

Question: Compare the distribution with the first dataset. What do you find?

Again you need to get the X and y out the dataframe and do train test split.

In [ ]:
```python
# Get X and y from dataset
...
```

In [ ]:
```python
# Do train test split with random_state=0 and test_size=0.5
...
```

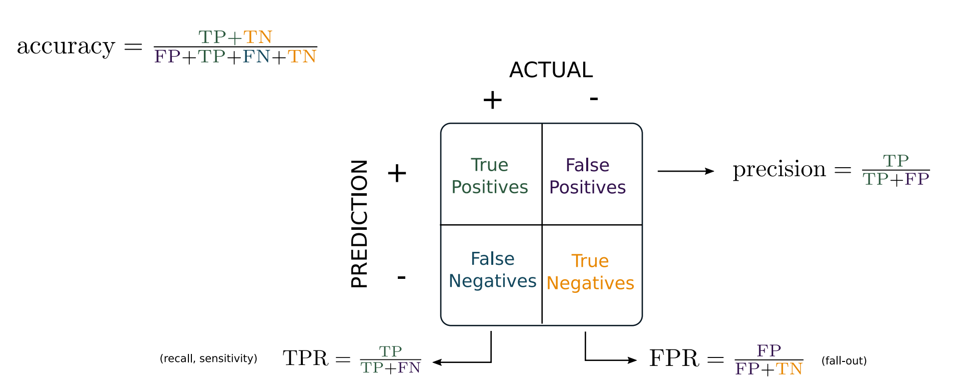
2.2.1 Accuracy for Naive Bayes

Let's train a Naive Bayes model again. Similar with 2.1.1, choose a proper Naive Bayes class with a proper smoothing variable to solve this priblem. Report your test accuracy.

In [ ]:
```python
...
```

Question: According to your accuracy score, how would you evaluate your model?

2.2.2 Confusion Matrix and Different Metrics



A very important tool to debug classifiers is the confusion matrix. For binary classification, it contains four different cells:

- **True positives (TP)**: observations that were predicted as belonging to the positive class correctly.
- **False positives (FP)**: observations that were predicted as belonging to the positive class incorrectly.
- **True negatives (TN)**: observations that were predicted as belonging to the negative class correctly.
- **False negatives (FN)**: observations that were predicted as belonging to the negative class incorrectly.

These are all interesting in and of themselves, but they can also be combined in aggregate metrics such as:

- **Accuracy**: how often are we predicting the class label?
- **Precision**: how many of our positive outcomes are actually positive?
- **Recall/Sensitivity**: how many of the positive outcomes are we able to recall?
- **Fall-Out**: how many of our negative outcomes are actually positive?

Suppose each data point in the data set represents a patient, and the class 1 represensts a patient is tested positive for a desease while 0 means tested negative. Choose a metric and report the metric score you choose for your model.

In [ ]:
```python
# Draw your confusion matrix here
...
```

In [ ]:
```python
# Calculate the metric you chose here
...
```

Question: Is the metric you chose higher is better or lower is better in this situation? What would you say about your model using the metric you chose?

In [ ]:
```python

```