Spam Classification

Goup1

Tian Yuhang(Timo)

16723065@lucbjtu.ac.uk

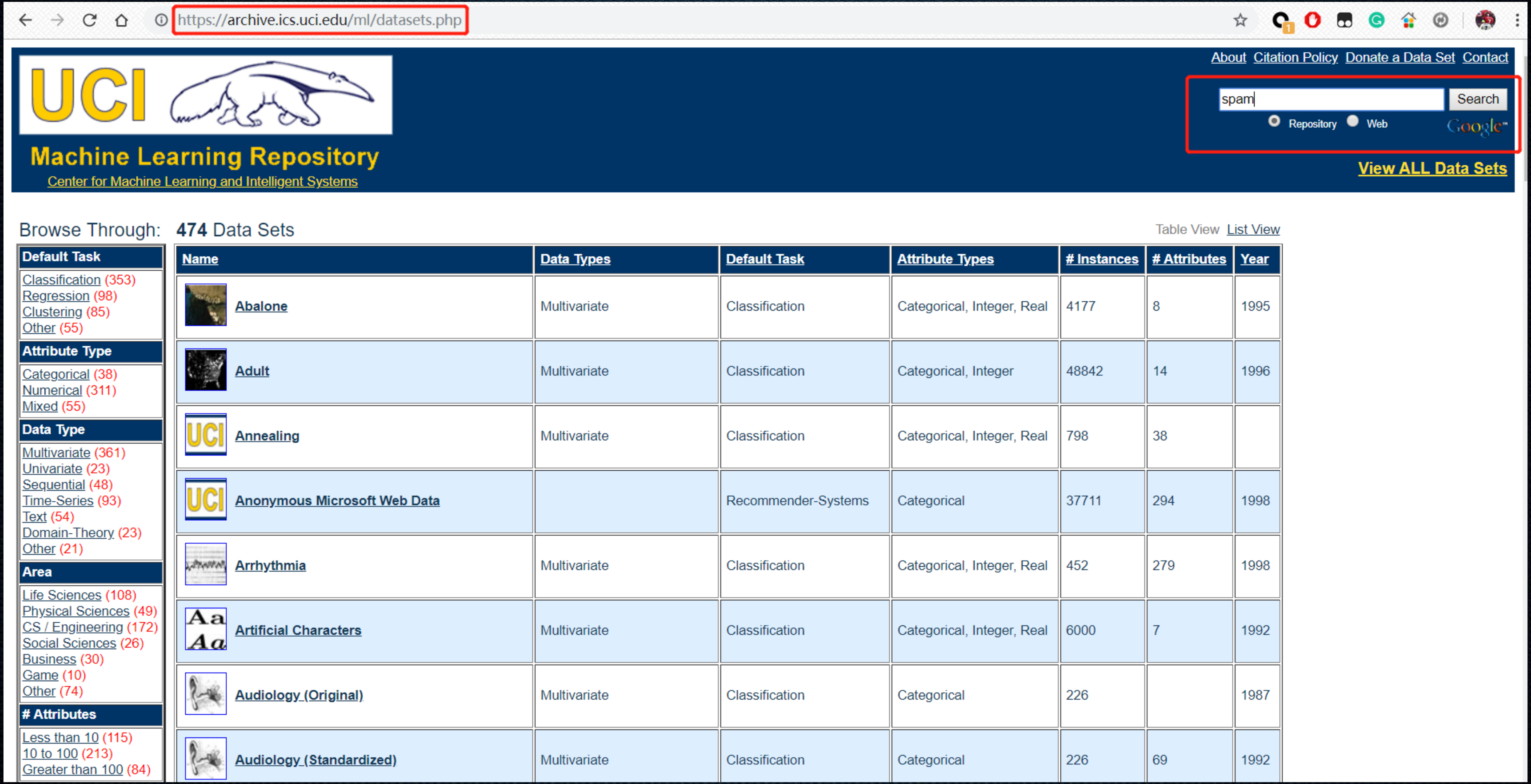
Content

- The place to find a dataset
- The operations of the dataset
- The application of KNN
- The estimation of model

1

The place to find a dataset

The place to find a dataset



https://archive.ics.uci.edu/ml/datasets.php

About Citation Policy Donate a Data Set Contact

spam Search

Repository Web Google

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Browse Through: **474** Data Sets

Table View List View

Default Task

- Classification (353)
- Regression (98)
- Clustering (85)
- Other (55)

Attribute Type

- Categorical (38)
- Numerical (311)
- Mixed (55)

Data Type

- Multivariate (361)
- Univariate (23)
- Sequential (48)
- Time-Series (93)
- Text (54)
- Domain-Theory (23)
- Other (21)

Area

- Life Sciences (108)
- Physical Sciences (49)
- CS / Engineering (172)
- Social Sciences (26)
- Business (30)
- Game (10)
- Other (74)

 # Attributes

- Less than 10 (115)
- 10 to 100 (213)
- Greater than 100 (84)

Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
Audiology (Original)	Multivariate	Classification	Categorical	226		1987
Audiology (Standardized)	Multivariate	Classification	Categorical	226	69	1992

The place to find a dataset

Spambase Data Set

Download: Data Folder, Data Set Description

Abstract: Classifying Email as Spam or Non-Spam

Data Set Characteristics:	Multivariate	Number of Instances:	4601	Area:	Computer
Attribute Characteristics:	Integer, Real	Number of Attributes:	57	Date Donated	1999-07-01
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	423033

Source:

Creators:

Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt
Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304

Donor:

George Forman (gforman at nospam hpl.hp.com) 650-857-7835

Data Set Information:

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography...

Our collection of spam e-mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

For background on spam:

Cranor, Lorrie F., LaMacchia, Brian A. Spam!
Communications of the ACM, 41(8):74-83, 1998.

(a) Hewlett-Packard Internal-only Technical Report. External forthcoming.
(b) Determine whether a given email is spam or not.
(c) ~7% misclassification error. False positives (marking good mail as spam) are very undesirable.If we insist on zero false positives in the training/testing set, 20-25% of the spam passed through the filter.

2 The operations of the dataset

The operations of the dataset

Spam Classification

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

word_freq_WORD	char_freq_CHAR	capital_run_length_average	capital_run_length_longest	capital_run_length_tota	spam
0.000	0.000	3.756	61	278	1
0.180	0.048	5.114	101	1028	1
0.184	0.010	9.821	485	2259	1
0.000	0.000	3.537	40	191	1
0.000	0.000	3.537	40	191	1

In [2]:

```python
# define column names
names = ['word_freq_WORD ', 'char_freq_CHAR', 'capital_run_length_average', 'capital_run_length_longest', 'capital_run_length_tota','spam']
```

48 continuous real [0,100] attributes of type word_freq_WORD = percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.

6 continuous real [0,100] attributes of type char_freq_CHAR] = percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,...] attribute of type capital_run_length_average = average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_longest = length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_total = sum of length of uninterrupted sequences of capital letters = total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam = denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

In [3]:

```python
# import the dataset
df = pd.read_csv('spambase.data', header=None,names=names)
df.head()
```

The operations of the dataset

Split and Train the Data

In [4]:

```
#split the dataset
scaler = StandardScaler()
scaler.fit(df.drop('spam', axis=1))

scaled_features = scaler.transform(df.drop('spam', axis=1))

df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()
```

	word_freq_WORD	char_freq_CHAR	capital_run_length_average	capital_run_length_longest	capital_run_length_tota
0	-0.308355	-0.103048	-0.045247	0.045298	-0.008724
1	0.423783	0.008763	-0.002443	0.250563	1.228324
2	0.440053	-0.079754	0.145921	2.221106	3.258733
3	-0.308355	-0.103048	-0.052150	-0.062466	-0.152222
4	-0.308355	-0.103048	-0.052150	-0.062466	-0.152222

In [5]:

```
# train the dataset
X = df_feat
y = df['spam']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
accuracy = knn.score(X_train,y_train)
print(accuracy)
```

```
0.8684782608695653
```

3

The application of KNN

The application of KNN

Preditction

In [6]:

```
# do the prediction
pred = knn.predict(X_test)
print(confusion_matrix(y_test, pred))
print('\n')
print(classification_report(y_test, pred))
```

```
[[502  59]
 [ 95 265]]


              precision    recall  f1-score   support

           0       0.84      0.89      0.87       561
           1       0.82      0.74      0.77       360

   micro avg       0.83      0.83      0.83       921
   macro avg       0.83      0.82      0.82       921
weighted avg       0.83      0.83      0.83       921
```

The application of KNN

In [7]:

```
y_true = [0, 1, 2, 2, 2]
y_pred = [0, 0, 2, 2, 1]
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

   micro avg       0.60      0.60      0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```

Precision: How many selected items are relevant?
Recall: How many relevant items are selected?
F1: The Harmonic Mean of precision and recall.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Real	Predict
0	0
1	0
2	2
2	2
2	1

	precision	recall	f1-score	support
0	0.50	1.00	0.67	1
1	0.00	0.00	0.00	1
2	1.00	0.67	0.80	3

The application of KNN

In [8]:

```
example1 = np.array([0.000,0.000,3.756,61,278])
example1 = example1.reshape(1,-1)
prediction = knn.predict(example1)
print(prediction)
```

```
[1]
```

4 The estimation of model

The estimation of model

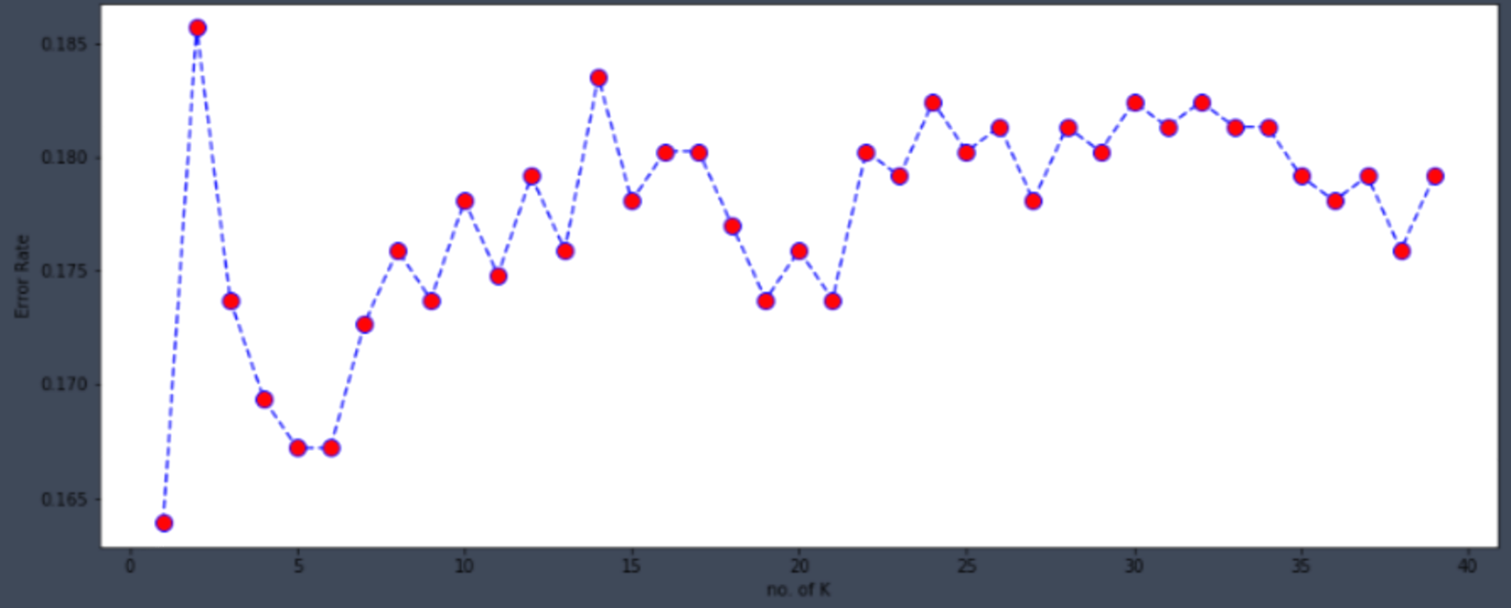
How to choose an appropriate K value?

In [9]:

```
error_rate = []
for i in range(1,40):
    knn = KNeighborsClassifier(i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [10]:

```
plt.figure(figsize=(15,6))
plt.plot(range(1,40),error_rate,color='blue',linestyle='dashed',marker='o', markerfacecolor='red', markersize='10')
plt.xlabel('no. of K')
plt.ylabel('Error Rate')
```



The estimation of model

In [11]:

```
knn = KNeighborsClassifier(5)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[502  59]
 [ 95 265]]
              precision    recall  f1-score   support

           0       0.84      0.89      0.87       561
           1       0.82      0.74      0.77       360

   micro avg       0.83      0.83      0.83       921
   macro avg       0.83      0.82      0.82       921
weighted avg       0.83      0.83      0.83       921
```

VS

In [12]:

```
knn = KNeighborsClassifier(8)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[517  44]
 [118 242]]
              precision    recall  f1-score   support

           0       0.81      0.92      0.86       561
           1       0.85      0.67      0.75       360

   micro avg       0.82      0.82      0.82       921
   macro avg       0.83      0.80      0.81       921
weighted avg       0.83      0.82      0.82       921
```

References

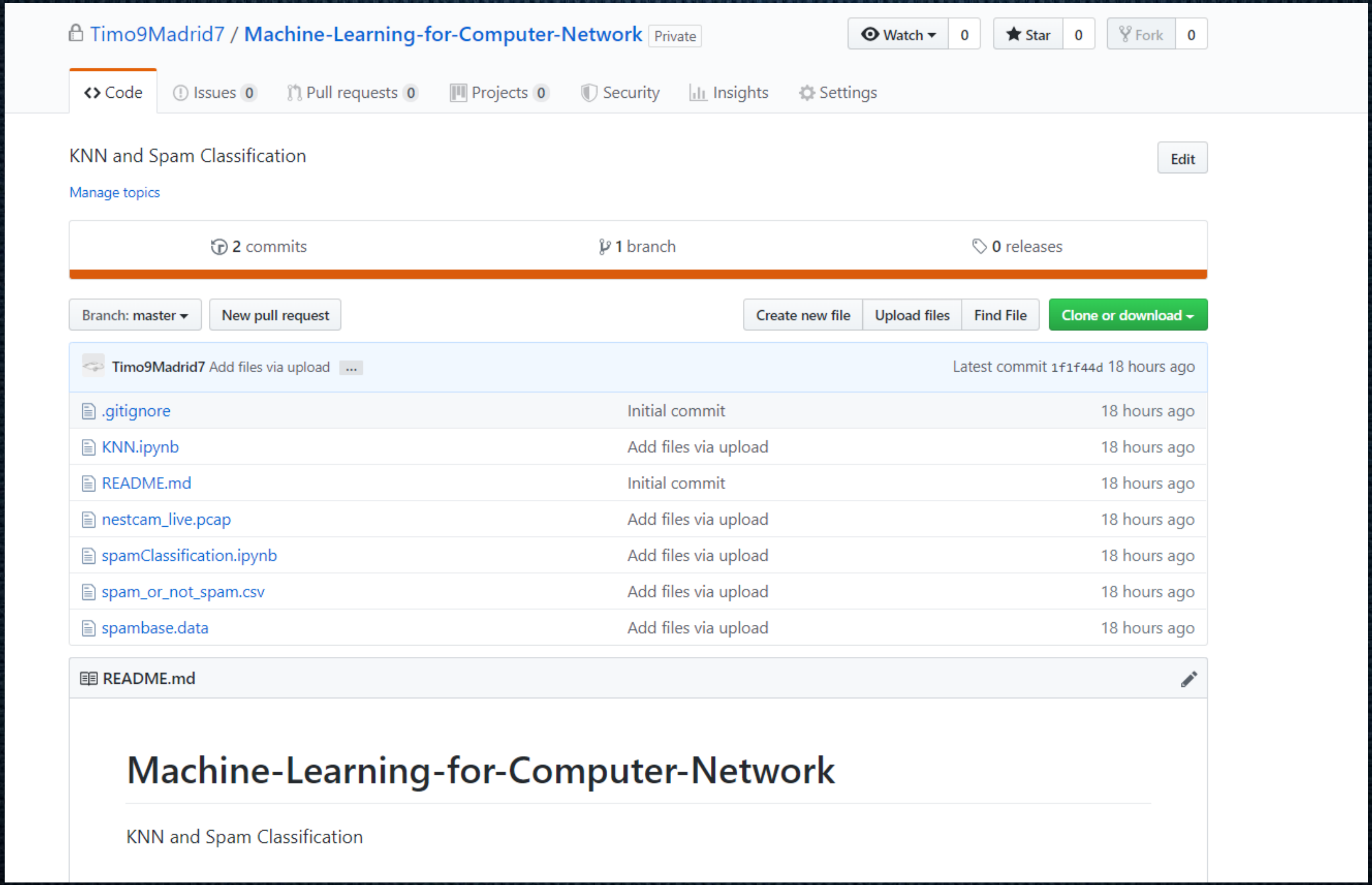
https://github.com/shoaibb/K-Nearest-Neighbors/blob/master/K-Nearest%20Neighbors.ipynb

https://github.com/NoahApthorpe/AI4All-IoT/blob/master/6_Nearest_Neighbors_sol.ipynb

https://archive.ics.uci.edu/ml/datasets/spambase

https://pythonprogramming.net/machine-learning-tutorial-python-introduction/

https://github.com/Timo9Madrid7/Machine-Learning-for-Computer-Network



Question?

Tian Yuhang
16723065@lucbjtu.ac.uk

Thanks for Watching!