Some tips for exercise set 2

0. Reading the data:

   - R:

   ```
   d <- read.csv('http://coltekin.net/cagri/courses/ml/data/fake-twitter-data.csv')
   ```

   - Python:

   ```
   import pandas as pd
   d = pd.read_csv('http://coltekin.net/cagri/courses/ml/data/fake-twitter-data.csv')
   ```

1. Simple logistic regression fit

   - R

   ```
   m <- glm(gender ~  sent_len, data=d, family=binomial)
   summary(m)
   pred <- predict(m, newdata=data.frame(sent_len=c(10, 50, 100, 140)),
           type='response')
   pred    # this gives probabilities
   # get the labels and adjust the index to produce labels from the prob.
   levels(d$gender)[as.integer(pred > 0.5]) + 1]
   ```

   - Python

   ```
   from sklearn.linear_model import LogisticRegression
   import numpy as np

   m = LogisticRegression()
   m.fit(d[['sent_len']], d['gender'])
   m.intercept_
   m.coef_
   m.predict(np.matrix('10; 50; 100; 140'))
   ```

2. Multiple predictors

   The commands that are same/similar to above are skipped

   - R

   ```
   m <- glm(gender ~  word_len, sent_len, data=d, family=binomial)
   predict(m, newdata=data.frame(sent_len=c(50, 50, 100, 100),
                                 word_len=c(2, 5, 2, 5)),
           type='response')
   ```

   - Python

   ```
   m.fit(d[['sent_len', 'word_len']], d['gender'])
   m.predict(np.matrix('50, 2; 50, 5; 100, 2; 100, 5'))
   ```

3. Accuracy

- R:

```
predicted <- as.integer(predict(m, type='response') > 0.5)
expected <- as.integer(d$gender) - 1
sum(pred == expected) / length(d$gender)
```

- Python: using `sklearn.metrics` is a better idea, but here is manual calculation for the sake of demonstration

```
float(np.sum(m.predict(d[['sent_len', 'word_len']]) == d['gender'])) / len(d)
```

4. Training/test set

   Only splitting the data, the rest is similar to above

   - R

```
train <- d[1:(dim(d)[1]/2),]
test <- d[(dim(d)[1]/2 + 1):dim(d)[1],]
```

   - Python

```
train = d[:len(d)/2+1]
test = d[len(d)/2+1:]
```

5. Plot the sigmoid

   Assming the model fitted in **exercise 1** is saved as `m`

   - R:

```
plot(d$sent_len, as.integer(d$gender) - 1,
     col=c('blue', 'red')[d$gender])
x <- min(d$sent_len):max(d$sent_len)
y <- predict(m, newdata=data.frame(sent_len=x), type='response')
lines(x,y)
```

   - Python: can be much better, but just to show the possibilities

```
import matplotlib.pyplot as plt

M = d[d.gender == 'M']
F = d[d.gender == 'F']
plt.axis([min(d.sent_len), max(d.sent_len), 0, 1])
plt.plot(F.sent_len, [1 for i in F.sent_len], 'ro')
plt.plot(M.sent_len, [0 for i in M.sent_len], 'bo')
x = range(min(d.sent_len), max(d.sent_len))
y = m.predict_proba(np.matrix(x).transpose()).transpose()[1]
plt.plot(x, y)
plt.show()
```