In [2]:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn.metrics as metrics
import seaborn as sn
drRatings = pd.read_excel('./OBGYN_new_train_80000.xlsx')
#shuffle the data so that they are in random sequence  #?
drRatings = drRatings.sample(frac=1)
drRatings['highPunctuality'] = (drRatings['punctuality']>4).astype(int)
```

In [3]:
```
drRatingsRatings
```

Out[3]:

	reviewID	doctorID	doctorName	specialty	numReviews	city	state	doctorHomepage	averageRating	staff	punctual
76000	76001	90457	Dr. Romy E. Mason	Gynecologist (OBGYN)	9	Denver	CO	/doctor-ratings/90457/Dr-Romy%2BE.-Mason-Denve...	5.00	5	
8417	8418	1656	Dr. Samina Raghid	Gynecologist (OBGYN)	29	New York	NY	/doctor-ratings/1656/Dr-Samina-Raghid-New%2BYo...	3.25	1	
12815	12816	665359	Dr. Victor J. Weinstein	Gynecologist (OBGYN)	4	Charleston	SC	/doctor-ratings/665359/Dr-Victor%2BJ.-Weinstei...	5.00	5	
25751	25752	35420	Dr. Charles A. Bryz-Gornia	Gynecologist (OBGYN)	15	Maple Grove	MN	/doctor-ratings/35420/Dr-Charles%2BA.-Bryz-Gor...	1.50	3	
38499	38500	169011	Dr. Steven Hockstein	Gynecologist (OBGYN)	13	New York	NY	/doctor-ratings/169011/Dr-Steven-Hockstein-New...	5.00	5	
...	...	...	...	...	...	...	...	...	...	...	
1320	1321	141186	Dr. Molly McBride	Gynecologist (OBGYN)	23	WILMINGTON	DE	/doctor-ratings/141186/Dr-Molly-McBride-WILMIN...	2.00	1	
18737	18738	202323	Dr. Amy D. Greenwald	Gynecologist (OBGYN)	3	Jacksonville	FL	/doctor-ratings/202323/Dr-Amy%2BD.-Greenwald-J...	4.50	5	
9925	9926	489296	Dr. Madeline Rodriguez	Gynecologist (OBGYN)	2	Oceanside	CA	/doctor-ratings/489296/Dr-Madeline-Rodriguez-O...	5.00	5	
23573	23574	2459	Dr. Nancy J. Bohannon	Gynecologist (OBGYN)	9	Silverdale	WA	/doctor-ratings/2459/Dr-Nancy%2BJ.-Bohannon-Si...	5.00	0	

	reviewID	doctorID	doctorName	specialty	numReviews	city	state	doctorHomepage	averageRating	staff	punctual
15168	15169	3235784	Dr. Jessica D. Close	Gynecologist (OBGYN)	1	Hagerstown	MD	/doctor-ratings/3235784/Dr-JESSICA%2BD.-CLOSE-...	4.50	4	

80000 rows × 17 columns

In [2]:

```
temp = pd.get_dummies(drRatings['state']) #?
drRatings = pd.concat([drRatings,temp],axis=1)
del temp
drRatings['postedTime']=pd.to_datetime(drRatings['postedTime'])
drRatings['year']=drRatings['postedTime'].dt.year
drRatings['hour']=drRatings['postedTime'].dt.hour
```

In [4]:

```
import re
from sklearn import feature_extraction
stop_words = feature_extraction.text.ENGLISH_STOP_WORDS
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

def preprocess(text):
  text = text.lower() #lowercase
  text = re.sub(r'[^\w\s]', '', text) #remove punctuations
  text = re.sub(r'\d+', '', text) #remove numbers
  text = " ".join(text.split()) #stripWhitespace
  text = text.split()
  text = [x for x in text if x not in stop_words] #remove stopwords
  text = [x for x in text if x not in ["dr", "doctor"]] #remove task specific stopwords
  text = " ".join(text)
  # stemmer_ps = PorterStemmer()
  # text = [stemmer_ps.stem(word) for word in text.split()] #stemming
  # text = " ".join(text)
  # lemmatizer = WordNetLemmatizer()
  # text = [lemmatizer.lemmatize(word) for word in text.split()]  #lemmatization
  # text = " ".join(text)
  return(text)
```

In [46]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import feature_extraction
stop_words = feature_extraction.text.ENGLISH_STOP_WORDS
stop_words = ["dr", "doctor"] + list(stop_words)

def calTFIDF(texts,max_features=None):
  vectorizer = TfidfVectorizer(max_features=max_features,lowercase=True,stop_words=stop_words,ngram_rang
  TFIDF = vectorizer.fit_transform(texts)
  TFIDF=pd.DataFrame(TFIDF.toarray(),columns=vectorizer.get_feature_names())
  return(TFIDF)
```

In [47]:

```
drRatings['text'] = drRatings['review'].apply(lambda x:preprocess(x))
```

In [48]:
```
TFIDF=calTFIDF(drRatings['text'],max_features=250)
TFIDF
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_fe
ature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please u
se get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[48]:

	able	absolutely	actually	ago	amazing	answer	answered	answers	appointment	appointments	...	woman	women	won
0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.0	0.0
1	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.0	0.0
2	0.000000	0.0	0.213766	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.0	0.1
3	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.270141	0.000000	0.0	...	0.0	0.0	0.1
4	0.267559	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.208431	0.0	...	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
79995	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.380786	0.000000	0.0	...	0.0	0.0	0.0
79996	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.0	0.0
79997	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.0	0.0
79998	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.0	0.0
79999	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.231035	0.0	...	0.0	0.0	0.0

80000 rows × 250 columns

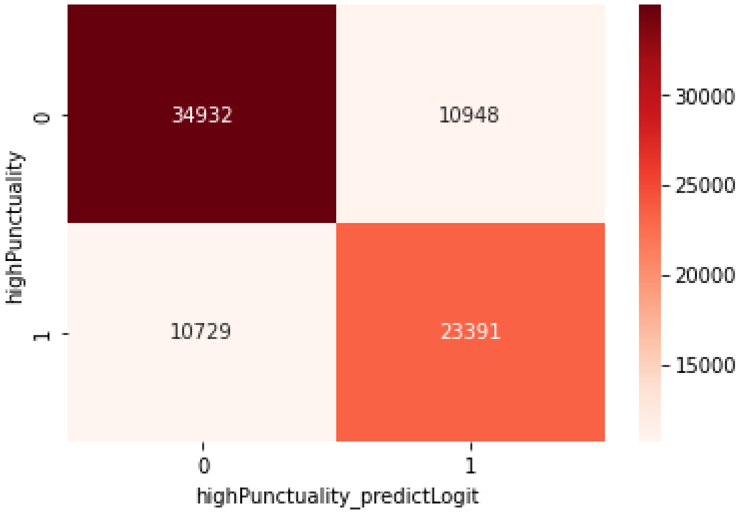
In [49]:

```
xcols = ['AK', 'AL', 'AR', 'AZ', 'CA','CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN',
       'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'NC', 'ND', 'NE',
       'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'PR', 'RI', 'SC', 'SD',
       'TN', 'TX', 'UT', 'VA', 'WA', 'WI', 'WV', 'WY', 'year', 'hour','numReviews']

# drRatings = pd.concat([drRatings,TFIDF],axis=1)
ycol = 'highPunctuality'
x = drRatings[xcols]
x = sm.add_constant(x)
x = np.concatenate([x,TFIDF.values],axis=1)
y = drRatings[ycol]

logit_model1 = sm.Logit(y, x)
logit_result = logit_model1.fit()

drRatings['highPunctuality_predictLogit'] = (logit_result.predict(x) >= 0.5).astype(int)
acc=metrics.accuracy_score(y_true=drRatings['highPunctuality'],y_pred=drRatings['highPunctuality_predict
print('prediction accuracy is',acc)
confusion=metrics.confusion_matrix(y_true=drRatings['highPunctuality'],y_pred=drRatings['highPunctuality
# print(confusion)
sn.heatmap(confusion, annot=True, cmap='Reds', fmt='d')
plt.xlabel("highPunctuality_predictLogit")
plt.ylabel("highPunctuality")
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:117: FutureWarning: In a future ver
sion of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)

Optimization terminated successfully.
         Current function value: 0.528670
         Iterations 7
prediction accuracy is 0.7290375
```

Out[49]:

```
Text(33.0, 0.5, 'highPunctuality')
```



prediction on testing data (out of sample prediction)

In [50]:

```
xcols = ['AK', 'AL', 'AR', 'AZ', 'CA','CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN',
       'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'NC', 'ND', 'NE',
       'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'PR', 'RI', 'SC', 'SD',
       'TN', 'TX', 'UT', 'VA', 'WA', 'WI', 'WV', 'WY', 'year', 'hour','numReviews']

ycol = 'highPunctuality'
x = drRatings[xcols].values
x = sm.add_constant(x)
x = np.concatenate([x,TFIDF.values],axis=1)
y = drRatings[ycol]


x_train=x[:10000]
x_test=x[10000:]
y_train=y[:10000]
y_test=y[10000:]
drRatings=drRatings.reset_index(drop=True)
drRatings_train=drRatings.loc[:9999]
drRatings_train=drRatings_train.reset_index(drop=True)
drRatings_test=drRatings.loc[10000:]
drRatings_test=drRatings_test.reset_index(drop=True)

logit_model1 = sm.Logit(y_train, x_train)
logit_result = logit_model1.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.516565
         Iterations 7
```

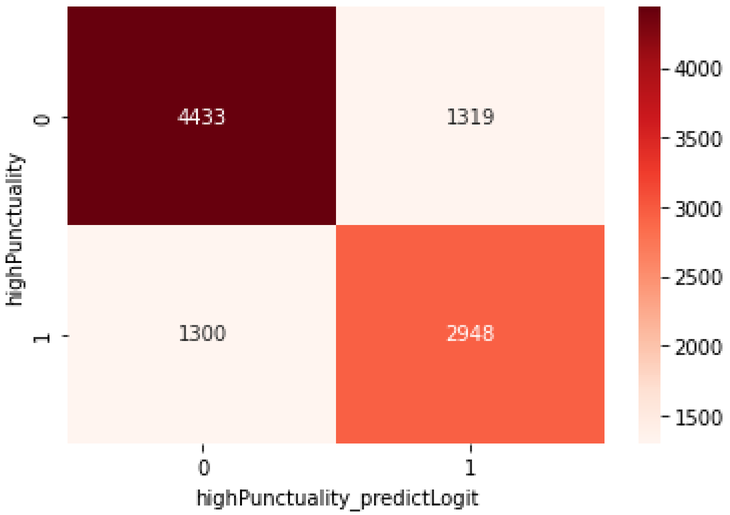
In [51]:

```
drRatings_train['highPunctuality_predictLogit'] = (logit_result.predict(x_train) >= 0.5).astype(int).tol
acc=metrics.accuracy_score(y_true=drRatings_train['highPunctuality'],y_pred=drRatings_train['highPunctua
print('prediction accuracy is',acc)
confusion=metrics.confusion_matrix(y_true=drRatings_train['highPunctuality'],y_pred=drRatings_train['hig
# print(confusion)
sn.heatmap(confusion, annot=True, cmap='Reds', fmt='d')
plt.xlabel("highPunctuality_predictLogit")
plt.ylabel("highPunctuality")
```

```
prediction accuracy is 0.7381
```

Out[51]:

```
Text(33.0, 0.5, 'highPunctuality')
```



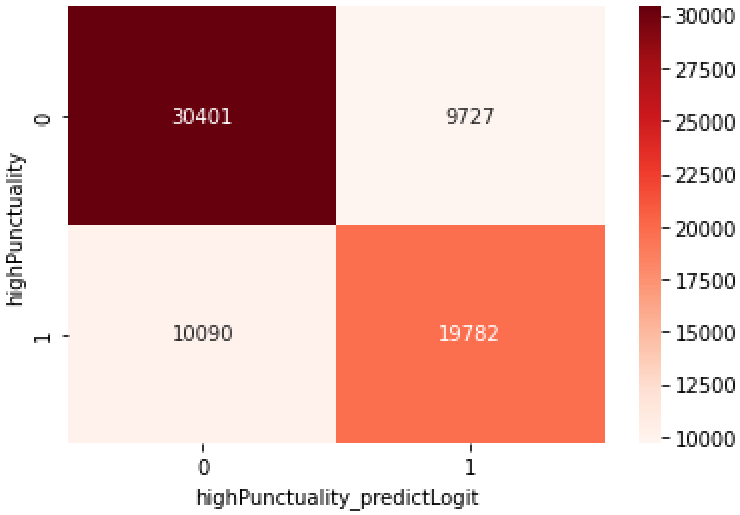
In [52]:

```
drRatings_test['highPunctuality_predictLogit'] = (logit_result.predict(x_test) >= 0.5).astype(int)
acc=metrics.accuracy_score(y_true=drRatings_test['highPunctuality'],y_pred=drRatings_test['highPunctuali
print('prediction accuracy is',acc)
confusion=metrics.confusion_matrix(y_true=drRatings_test['highPunctuality'],y_pred=drRatings_test['highP
# print(confusion)
sn.heatmap(confusion, annot=True, cmap='Reds', fmt='d')
plt.xlabel("highPunctuality_predictLogit")
plt.ylabel("highPunctuality")
```

```
prediction accuracy is 0.7169
```

Out[52]:

```
Text(33.0, 0.5, 'highPunctuality')
```



In [ ]:

In [ ]:

In [ ]:

prediction on testing data (out of sample prediction) using two files: OBGYN_new_train_80000.xlsx and OBGYN_new_test_lab_withAnswer_100.xlsx

this is how you create the submission and how TAs will calculate your accuracy

In [53]:

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn.metrics as metrics
import seaborn as sn
drRatings = pd.read_excel('./OBGYN_new_train_80000.xlsx')
#shuffle the data so that they are in random sequence
drRatings = drRatings.sample(frac=1)
drRatings['highPunctuality'] = (drRatings['punctuality']>4).astype(int)
```

In [54]:

```
testingdata = pd.read_excel('./OBGYN_new_test_lab_withAnswer_100.xlsx')
```

In [55]:

```
temp = pd.get_dummies(drRatings['state'])
drRatings = pd.concat([drRatings,temp],axis=1)
del temp
drRatings['postedTime']=pd.to_datetime(drRatings['postedTime'])
drRatings['year']=drRatings['postedTime'].dt.year
drRatings['hour']=drRatings['postedTime'].dt.hour
```

In [56]:

```
temp = pd.get_dummies(testingdata['state'])
testingdata = pd.concat([testingdata,temp],axis=1)
del temp
states=drRatings['state'].unique().tolist()
for state in states:
  if state not in testingdata.columns.tolist():
    testingdata[state]=[0]*testingdata.shape[0]
testingdata['postedTime']=pd.to_datetime(testingdata['postedTime'])
testingdata['year']=testingdata['postedTime'].dt.year
testingdata['hour']=testingdata['postedTime'].dt.hour
```

In [57]:

```
import re
from sklearn import feature_extraction
stop_words = feature_extraction.text.ENGLISH_STOP_WORDS
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

def preprocess(text):
  text = text.lower() #lowercase
  text = re.sub(r'[^\w\s]', '', text) #remove punctuations
  text = re.sub(r'\d+', '', text) #remove numbers
  text = " ".join(text.split()) #stripWhitespace
  text = text.split()
  text = [x for x in text if x not in stop_words] #remove stopwords
  text = [x for x in text if x not in ["dr", "doctor"]] #remove task specific stopwords
  text = " ".join(text)
  # stemmer_ps = PorterStemmer()
  # text = [stemmer_ps.stem(word) for word in text.split()] #stemming
  # text = " ".join(text)
  # lemmatizer = WordNetLemmatizer()
  # text = [lemmatizer.lemmatize(word) for word in text.split()]  #lemmatization
  # text = " ".join(text)
  return(text)
```

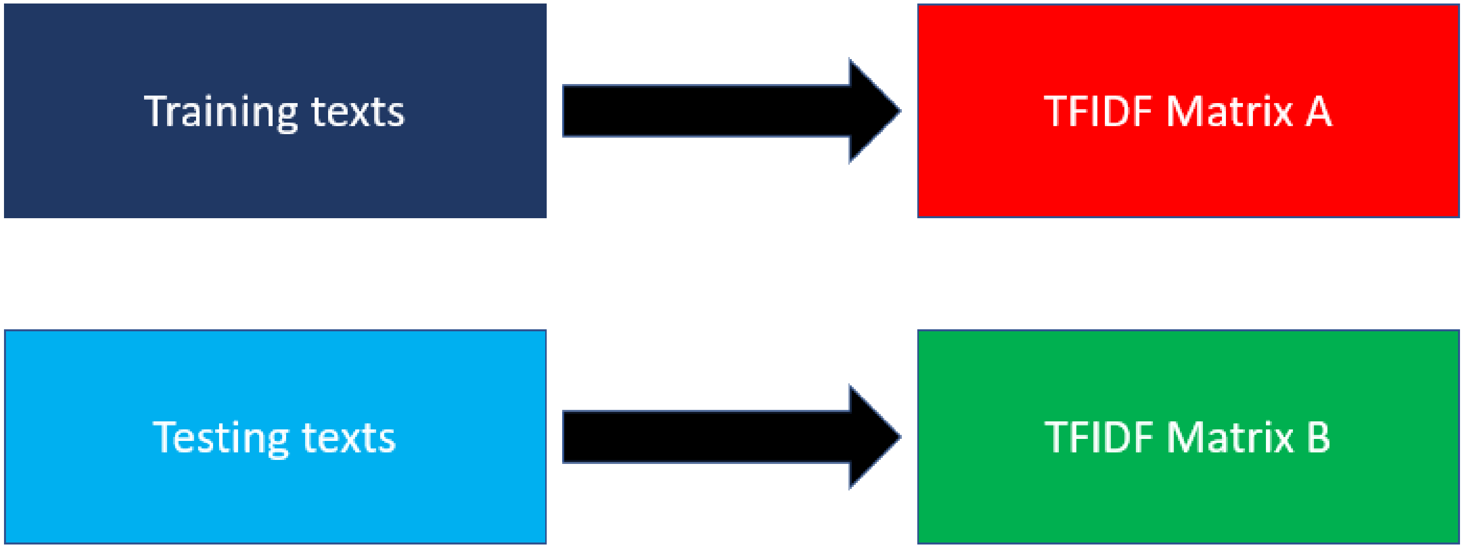
In [58]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import feature_extraction
stop_words = feature_extraction.text.ENGLISH_STOP_WORDS
stop_words = ["dr", "doctor"] + list(stop_words)

def calTFIDF(texts,max_features=None):
  vectorizer = TfidfVectorizer(max_features=max_features,lowercase=True,stop_words=stop_words,ngram_rang
  TFIDF = vectorizer.fit_transform(texts)
  TFIDF=pd.DataFrame(TFIDF.toarray(),columns=vectorizer.get_feature_names())
  return(TFIDF)
```

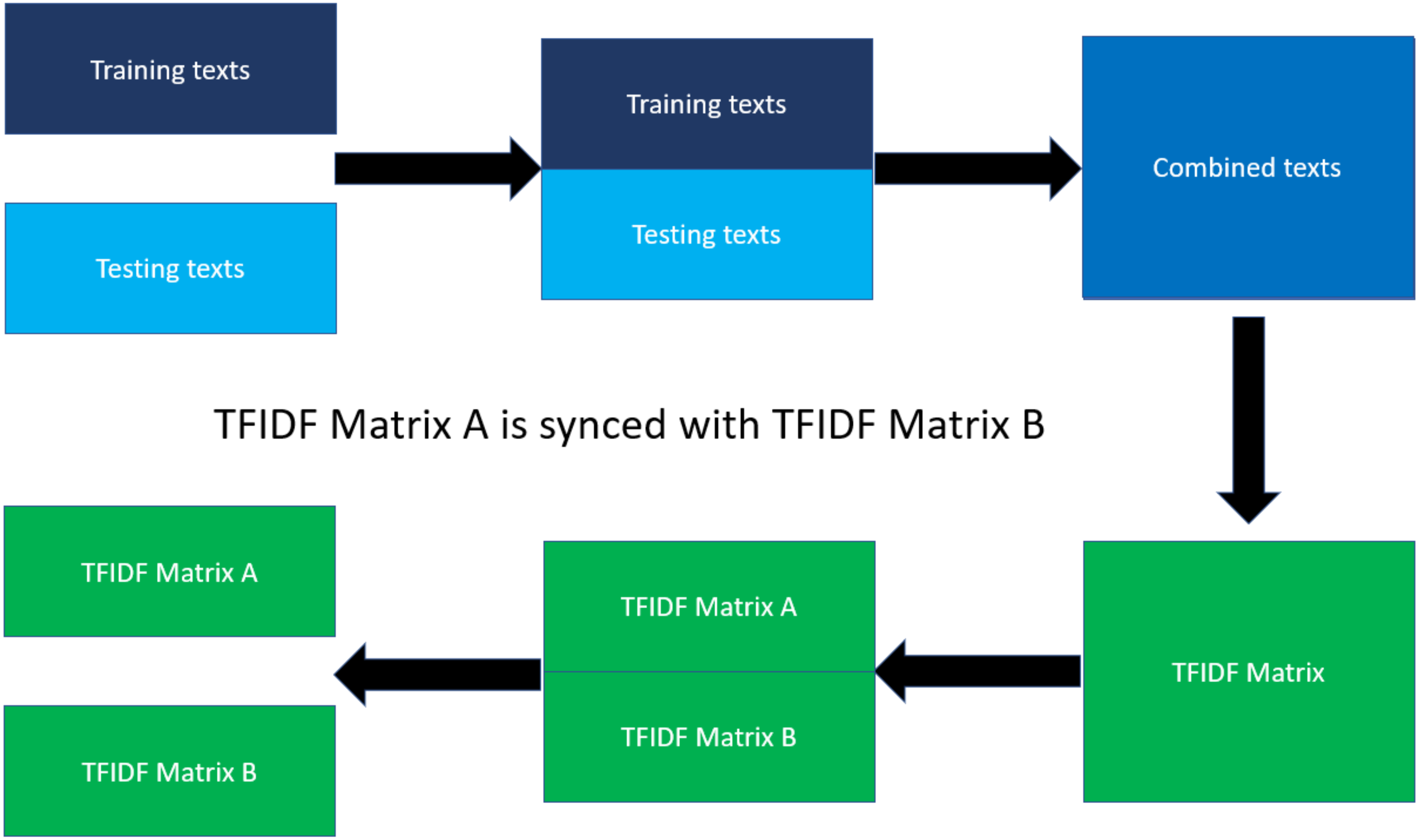
In [59]:

```
drRatings['text'] = drRatings['review'].apply(lambda x:preprocess(x))
testingdata['text'] = testingdata['review'].apply(lambda x:preprocess(x))
```

To understand the following few cells:



TFIDF Matrix A is **NOT** synced with TFIDF Matrix B



Training texts

Testing texts

Training texts

Testing texts

Combined texts

TFIDF Matrix A is synced with TFIDF Matrix B

TFIDF Matrix A

TFIDF Matrix B

TFIDF Matrix A

TFIDF Matrix B

TFIDF Matrix

```
In [60]: alltext=drRatings['text'].tolist()+testingdata['text'].tolist()
```

In [8]:

```
TFIDF=calTFIDF(alltext,max_features=250)
TFIDF.columns = ['tfidf_'+x for x in TFIDF.columns]
TFIDF_train = TFIDF[:80000]
TFIDF_test = TFIDF[80000:]
TFIDF_test=TFIDF_test.reset_index(drop=True)
TFIDF
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/cf/s1wshv2j2bz4cbfxfg5qgrgm0000gn/T/ipykernel_28931/4232235381.py in <module>
----> 1 TFIDF=calTFIDF(alltext,max_features=250)
      2 TFIDF.columns = ['tfidf_'+x for x in TFIDF.columns]
      3 TFIDF_train = TFIDF[:80000]
      4 TFIDF_test = TFIDF[80000:]
      5 TFIDF_test=TFIDF_test.reset_index(drop=True)

NameError: name 'calTFIDF' is not defined
```

In [7]:

```
xcols = ['AK', 'AL', 'AR', 'AZ', 'CA','CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN',
        'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'NC', 'ND', 'NE',
        'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'PR', 'RI', 'SC', 'SD',
        'TN', 'TX', 'UT', 'VA', 'WA', 'WI', 'WV', 'WY', 'year', 'hour','numReviews']

ycol = 'highPunctuality'
x = drRatings[xcols].values
x = sm.add_constant(x)
x = np.concatenate([x,TFIDF_train.values],axis=1)
y = drRatings[ycol]


logit_model1 = sm.Logit(y, x)
logit_result = logit_model1.fit()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/var/folders/cf/s1wshv2j2bz4cbfxfg5qgrgm0000gn/T/ipykernel_28931/379935943.py in <module>
      5
      6 ycol = 'highPunctuality'
----> 7 x = drRatings[xcols].values
      8 x = sm.add_constant(x)
      9 x = np.concatenate([x,TFIDF_train.values],axis=1)

~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/frame.py in __getitem__(self, key)
   3462             if is_iterator(key):
   3463                 key = list(key)
-> 3464             indexer = self.loc._get_listlike_indexer(key, axis=1)[1]
   3465
   3466         # take() does not accept boolean indexers

~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py in _get_listlike_indexer(self, ke
y, axis)
   1312             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)
   1313
-> 1314         self._validate_read_indexer(keyarr, indexer, axis)
   1315
   1316         if needs_i8_conversion(ax.dtype) or isinstance(

~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py in _validate_read_indexer(self, ke
y, indexer, axis)
   1375
   1376                 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 1377                 raise KeyError(f"{not_found} not in index")
   1378
   1379

KeyError: "['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL',
 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'NC', 'ND', 'NE', 'NJ', 'NM', 'NV',
'NY', 'OH', 'OK', 'OR', 'PA', 'PR', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'WA', 'WI', 'WV', 'WY',
 'year', 'hour'] not in index"
```

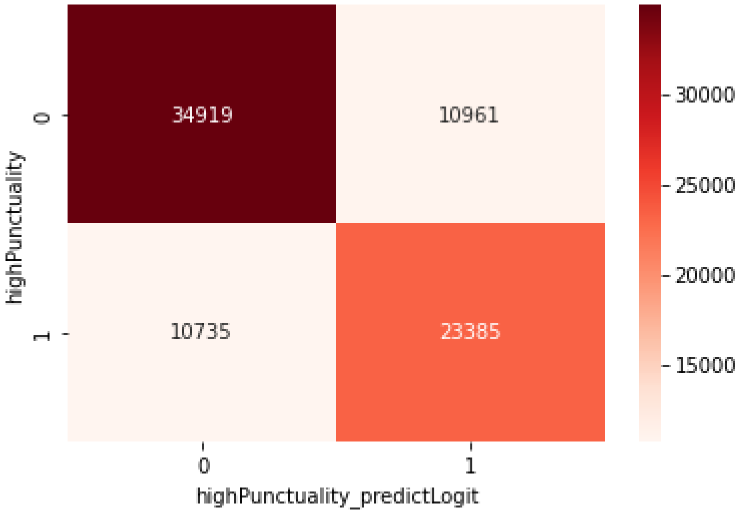
In [63]:

```
drRatings['highPunctuality_predictLogit'] = (logit_result.predict(x) >= 0.5).astype(int)
acc=metrics.accuracy_score(y_true=drRatings['highPunctuality'],y_pred=drRatings['highPunctuality_predict
print('prediction accuracy is',acc)
confusion=metrics.confusion_matrix(y_true=drRatings['highPunctuality'],y_pred=drRatings['highPunctuality
# print(confusion)
sn.heatmap(confusion, annot=True, cmap='Reds', fmt='d')
plt.xlabel("highPunctuality_predictLogit")
plt.ylabel("highPunctuality")
```

```
prediction accuracy is 0.7288
```

Out[63]:

```
Text(33.0, 0.5, 'highPunctuality')
```



In [65]:

```
xcols = ['AK', 'AL', 'AR', 'AZ', 'CA','CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN',
        'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'NC', 'ND', 'NE',
        'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'PR', 'RI', 'SC', 'SD',
        'TN', 'TX', 'UT', 'VA', 'WA', 'WI', 'WV', 'WY', 'year', 'hour','numReviews']

x = testingdata[xcols].values
x = sm.add_constant(x)
x = np.concatenate([x,TFIDF_test.values],axis=1)


pred = (logit_result.predict(x) >= 0.5).astype(int)
pred_prob = logit_result.predict(x)
```

In [66]:

```
print(pred)
print(pred_prob)
```

```
[0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1
 1 0 0 1 1 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0
 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 0]
[0.4238756  0.31937647 0.59725119 0.77731732 0.70973956 0.15962988
 0.07611441 0.88361983 0.40629449 0.53211333 0.04148109 0.25161243
 0.19463229 0.3607077  0.72022702 0.62500124 0.37692385 0.14530772
 0.29357838 0.06991137 0.78397747 0.85036535 0.31627968 0.3297796
 0.21787176 0.0456347  0.59218028 0.07044337 0.66475681 0.46255336
 0.66922551 0.20531901 0.60053502 0.46184112 0.8760508  0.30037022
 0.79401495 0.85914392 0.02460369 0.03155615 0.71194942 0.8729691
 0.69546288 0.04931416 0.87150199 0.11998477 0.72601605 0.11625639
 0.143124   0.04151834 0.54834568 0.5246423  0.75373502 0.1342557
 0.82934762 0.24965033 0.86030901 0.07123645 0.85548794 0.59853818
 0.37262093 0.11568658 0.57952671 0.45028569 0.03002278 0.34368463
 0.67439464 0.22401362 0.62002076 0.10241918 0.47468118 0.72498416
 0.19752417 0.49207785 0.34943949 0.44616309 0.25256363 0.17912831
 0.76274915 0.32606834 0.49311393 0.36954309 0.59061308 0.06643964
 0.04621719 0.12384819 0.62713336 0.91788141 0.39008042 0.00644224
 0.56926288 0.42917263 0.69580358 0.88645914 0.5096554  0.71757714
 0.58146098 0.68827712 0.04105898 0.44312255]
```

In [67]:

```
submitcsv=pd.DataFrame()
submitcsv['reviewID']=testingdata['reviewID'].tolist()
submitcsv['prediction']=pred
submitcsv.to_csv('submission.csv',index=False)
print(submitcsv)
```

```
    reviewID  prediction
0     100001           0
1     100002           0
2     100003           1
3     100004           1
4     100005           1
..       ...         ...
95    100096           1
96    100097           1
97    100098           1
98    100099           0
99    100100           0

[100 rows x 2 columns]
```

In [ ]:

In [ ]:

Below is how TAs will calculate your accuracy

In [68]:

```
testingdata = pd.read_excel('./OBGYN_new_test_lab_withAnswer_100.xlsx')
submission = pd.read_csv('./submission.csv')
testingdata['highPunctuality'] = (testingdata['punctuality']>4).astype(int)
acc=metrics.accuracy_score(y_true=testingdata['highPunctuality'].to_list(),y_pred=submission['prediction
print(acc)
confusion=metrics.confusion_matrix(y_true=testingdata['highPunctuality'],y_pred=pred)
print(confusion)
```

```
0.72
[[42 13]
 [15 30]]
```