Can timestamp and user activity be used to predict location on stackexchange?

April 25, 2019

In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime

import matplotlib.pyplot as plt # plots
import seaborn as sns

from sklearn.model_selection import train_test_split # splitting dataframes and subspl
from sklearn.ensemble import RandomForestClassifier

# evaluation metrics
from sklearn.metrics import classification_report # f1 scores
from sklearn.metrics import confusion_matrix # prediction vs actual
from sklearn.metrics import accuracy_score # overall score
from sklearn.model_selection import cross_val_score # check for overfitting and model

sns.set(style="darkgrid") # plotting if any
# plot formatting
%matplotlib inline
```

In [2]:
```
# this is for 50k users from each country with two timestamps except for nigeria, sout
# compare US to each of 7 other countries
# see if model can determine differences between these two countries
# based on timestamps
# hypothesis is users from different parts of world will have different timestamps

# steps
# step 1 - import csv files

# step 2 - clean dfs to get only datetime,
# change local to number based on key

# key
# 1: UK,
# 2: US,
# 3: CHINA,
```

```
        # 4: INDIA,
        # 5: RUSSIA,
        # 6: NIGERIA,
        # 7: SOUTHAFRICA,
        # 8: BRAZIL

        # use parsed and minute of week

        # step 3 - make functions to combine and shuffle

        # step 4 - modeling, functions here too,
        # fit,
        # test and get scores,

        # repeat steps 3&4 for each pair for total of 7
```

In [3]:
```
        # function to fit, predict, score for training and testing set
        def fit_pred_score(model, X_train, y_train, X_test, y_test):
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            class_report = (classification_report(y_test, y_pred))
            conf_mat = (confusion_matrix(y_test, y_pred))
            acc_score = (accuracy_score(y_test, y_pred))
            print(model)
            print(class_report)
            print(conf_mat)
            return acc_score
        def fit_pred_score_noprint(model, X_train, y_train, X_test, y_test):
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            class_report = (classification_report(y_test, y_pred))
            conf_mat = (confusion_matrix(y_test, y_pred))
            acc_score = (accuracy_score(y_test, y_pred))
            return acc_score
```

In [4]:
```
        # df's to combine, which countries
        # append
        # shuffle
        # get columns
        def pair_tobe_tested(df1, df2):
            dfname = pd.DataFrame()
            dfname = df1.append(df2)
            dfname = dfname.sample(frac=1).reset_index(drop=True)
            dfname = dfname[['activity', 'minute_of_week', 'minute_of_week2', 'local']]
            return dfname
```

In [5]:
```
        # import csv files
```

```
        # brazil.csv        india.csv        russia.csv        uk.csv
        # china.csv        nigeria.csv        southafrica.csv        us.csv
        df_uk = pd.read_csv('uk.csv')
        df_us = pd.read_csv('us.csv')
        df_chn = pd.read_csv('china.csv')
        df_ind = pd.read_csv('india.csv')
        df_rus = pd.read_csv('russia.csv')
        df_ng = pd.read_csv('nigeria.csv')
        df_sa = pd.read_csv('southafrica.csv')
        df_bra = pd.read_csv('brazil.csv')
```

In [6]:
```
df_list = [df_uk, df_us, df_chn, df_ind, df_rus, df_ng, df_sa, df_bra]
for df in df_list:
    print(df.shape)
```

```
(50000, 5)
(50000, 5)
(49629, 5)
(50000, 5)
(32430, 5)
(12740, 5)
(15865, 5)
(50000, 5)
```

In [7]:
```
df_rus.head()
```

Out[7]:

	id	location	LastAccessDate	\
0	4279	Moscow, Russia	2019-04-20 18:18:53	
1	125816	Moscow, Russia	2019-04-21 00:04:38	
2	2877241	Moscow, Russia	2019-04-17 16:36:24	
3	2319407	Russia, Saint-Petersburg	2019-04-21 05:13:16	
4	876298	Russia, Moscow	2019-04-20 21:09:41	

	CreationDate	activity
0	2008-09-02 16:10:17	27507
1	2009-06-19 15:31:19	23830
2	2013-10-13 23:45:02	20356
3	2013-04-25 11:02:08	16051
4	2011-08-03 09:28:48	14755

In [8]:
```
# add parsed and minute_of_week
for df in df_list:
    df['parsed'] = pd.to_datetime(df['LastAccessDate'], format='%Y-%m-%d %H:%M:%S')
    df['parsed2'] = pd.to_datetime(df['CreationDate'], format='%Y-%m-%d %H:%M:%S')
    df['minute_of_week'] = df['parsed'].apply(lambda row: (row.dayofweek * 24 * 60) +
    df['minute_of_week2'] = df['parsed2'].apply(lambda row: (row.dayofweek * 24 * 60)
```

In [9]:
```
# add local based on key
for df, i in zip(df_list, list(range(1,9))):
    df['local'] = i
```

```
In [10]: df_rus.head()
```

```
Out[10]:         id                  location       LastAccessDate  \
         0     4279            Moscow, Russia  2019-04-20 18:18:53
         1   125816            Moscow, Russia  2019-04-21 00:04:38
         2  2877241            Moscow, Russia  2019-04-17 16:36:24
         3  2319407  Russia, Saint-Petersburg  2019-04-21 05:13:16
         4   876298            Russia, Moscow  2019-04-20 21:09:41

                CreationDate  activity              parsed             parsed2  \
         0  2008-09-02 16:10:17     27507 2019-04-20 18:18:53 2008-09-02 16:10:17
         1  2009-06-19 15:31:19     23830 2019-04-21 00:04:38 2009-06-19 15:31:19
         2  2013-10-13 23:45:02     20356 2019-04-17 16:36:24 2013-10-13 23:45:02
         3  2013-04-25 11:02:08     16051 2019-04-21 05:13:16 2013-04-25 11:02:08
         4  2011-08-03 09:28:48     14755 2019-04-20 21:09:41 2011-08-03 09:28:48

            minute_of_week  minute_of_week2  local
         0            8298             2410      5
         1            8644             6691      5
         2            3876            10065      5
         3            8953             4982      5
         4            8469             3448      5
```

```
In [11]: # use random forest
         rfc = RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=10)
```

```
In [12]: # comparison between uk and us
         df_us_uk = pair_tobe_tested(df_us, df_uk)
         X = df_us_uk[['activity', 'minute_of_week', 'minute_of_week2']]
         y = df_us_uk['local']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
         acc_us_uk = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
         fit_pred_score(rfc, X_train, y_train, X_test, y_test)
         scores = cross_val_score(rfc, X_train, y_train, cv=10)
         print(scores, scores.mean(), scores.std())
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
              precision    recall  f1-score   support

           1       0.90      0.64      0.75     10000
           2       0.72      0.93      0.81     10000

   micro avg       0.78      0.78      0.78     20000
```

```
   macro avg       0.81      0.78      0.78     20000
weighted avg       0.81      0.78      0.78     20000

[[6353 3647]
 [ 693 9307]]
[0.78825  0.782375 0.780875 0.778125 0.782125 0.77925  0.783    0.783125
 0.78075  0.780625] 0.78185 0.002618324846156418
```

```
In [13]: df_us_uk.head()
```

Out[13]:

	activity	minute_of_week	minute_of_week2	local
0	46	3867	3769	2
1	44	2616	4062	2
2	21	6482	9442	1
3	9	8249	2025	1
4	401	8659	4630	2

```
In [14]: # comparison between us and china
         df_us_chn = pair_tobe_tested(df_us, df_chn)
         X = df_us_chn[['activity', 'minute_of_week', 'minute_of_week2']]
         y = df_us_chn['local']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
         acc_us_chn = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
         fit_pred_score(rfc, X_train, y_train, X_test, y_test)
         scores = cross_val_score(rfc, X_train, y_train, cv=10)
         print(scores, scores.mean(), scores.std())
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
              precision    recall  f1-score   support

           2       0.88      0.99      0.93     10000
           3       0.99      0.86      0.92      9926

   micro avg       0.92      0.92      0.92     19926
   macro avg       0.93      0.92      0.92     19926
weighted avg       0.93      0.92      0.92     19926

[[9912   88]
 [1412 8514]]
[0.92823987 0.92435077 0.92723623 0.93111669 0.92747804 0.92484316
 0.92459222 0.92283563 0.9229611  0.92923463] 0.9262888352964564 0.002642482377035641
```

```
In [15]: # comparison between us and india
         df_us_ind = pair_tobe_tested(df_us, df_ind)
         X = df_us_ind[['activity', 'minute_of_week', 'minute_of_week2']]
         y = df_us_ind['local']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
         acc_us_ind = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
         fit_pred_score(rfc, X_train, y_train, X_test, y_test)
         scores = cross_val_score(rfc, X_train, y_train, cv=10)
         print(scores, scores.mean(), scores.std())

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
              precision    recall  f1-score   support

           2       0.88      0.72      0.79     10000
           4       0.76      0.90      0.83     10000

   micro avg       0.81      0.81      0.81     20000
   macro avg       0.82      0.81      0.81     20000
weighted avg       0.82      0.81      0.81     20000

[[7234 2766]
 [ 998 9002]]
[0.808375 0.822625 0.81375  0.815125 0.8095   0.82125  0.817875 0.815875
 0.8135   0.815125] 0.8153 0.004281573892857649


In [16]: # comparison between us and russia
         df_us_rus = pair_tobe_tested(df_us, df_rus)
         X = df_us_rus[['activity', 'minute_of_week', 'minute_of_week2']]
         y = df_us_rus['local']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
         acc_us_rus = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
         fit_pred_score(rfc, X_train, y_train, X_test, y_test)
         scores = cross_val_score(rfc, X_train, y_train, cv=10)
         print(scores, scores.mean(), scores.std())

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
```

```
            warm_start=False)
              precision    recall  f1-score   support

           2       0.85      0.99      0.91     10000
           5       0.98      0.73      0.84      6486

   micro avg       0.89      0.89      0.89     16486
   macro avg       0.91      0.86      0.87     16486
weighted avg       0.90      0.89      0.88     16486

[[9883  117]
 [1752 4734]]
[0.89112964 0.88794541 0.88809704 0.89067475 0.89126479 0.88853503
 0.88155899 0.88368214 0.89065817 0.88201395] 0.887555992616717 0.0035911588880144065
```

```
In [17]: # comparison between us and nigeria
         df_us_ng = pair_tobe_tested(df_us, df_ng)
         X = df_us_ng[['activity', 'minute_of_week', 'minute_of_week2']]
         y = df_us_ng['local']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
         acc_us_ng = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
         fit_pred_score(rfc, X_train, y_train, X_test, y_test)
         scores = cross_val_score(rfc, X_train, y_train, cv=10)
         print(scores, scores.mean(), scores.std())
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
              precision    recall  f1-score   support

           2       0.97      1.00      0.98     10000
           6       1.00      0.87      0.93      2548

   micro avg       0.97      0.97      0.97     12548
   macro avg       0.98      0.93      0.96     12548
weighted avg       0.97      0.97      0.97     12548

[[9997    3]
 [ 337 2211]]
[0.97111554 0.97390438 0.97728631 0.97290297 0.97409843 0.97449691
 0.97409843 0.96772265 0.97011357 0.97409843] 0.972983761308621 0.00254539950809886
```

```
In [18]: # comparison between us and nigeria
```

```
        df_us_sa = pair_tobe_tested(df_us, df_sa)
        X = df_us_sa[['activity', 'minute_of_week', 'minute_of_week2']]
        y = df_us_sa['local']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
        acc_us_sa = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
        fit_pred_score(rfc, X_train, y_train, X_test, y_test)
        scores = cross_val_score(rfc, X_train, y_train, cv=10)
        print(scores, scores.mean(), scores.std())
```

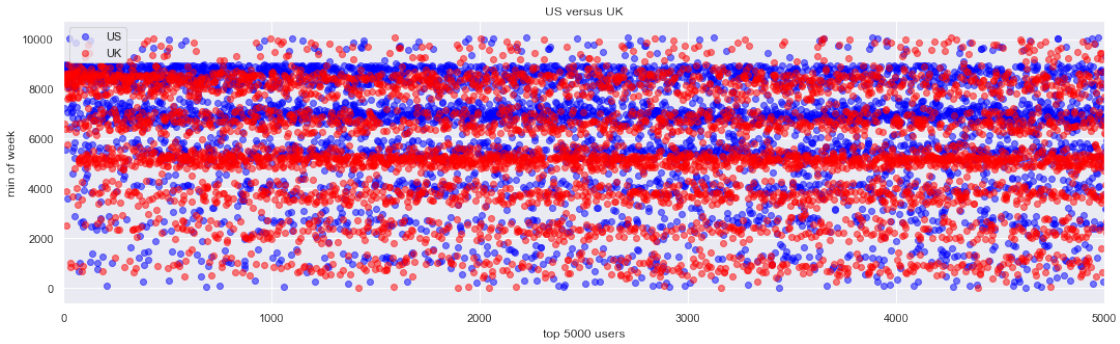
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
              precision    recall  f1-score   support

           2       0.93      1.00      0.96     10000
           7       0.99      0.78      0.87      3173

   micro avg       0.94      0.94      0.94     13173
   macro avg       0.96      0.89      0.92     13173
weighted avg       0.95      0.94      0.94     13173

[[9971   29]
 [ 704 2469]]
[0.94914611 0.94345351 0.93547163 0.94192446 0.94818751 0.93812868
 0.94097552 0.94097552 0.94363257 0.94761814] 0.942951364592513 0.004199457034335918
```

```
In [19]: # comparison between us and nigeria
        df_us_bra = pair_tobe_tested(df_us, df_bra)
        X = df_us_bra[['activity', 'minute_of_week', 'minute_of_week2']]
        y = df_us_bra['local']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
        acc_us_bra = fit_pred_score_noprint(rfc, X_train, y_train, X_test, y_test)
        fit_pred_score(rfc, X_train, y_train, X_test, y_test)
        scores = cross_val_score(rfc, X_train, y_train, cv=10)
        print(scores, scores.mean(), scores.std())
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

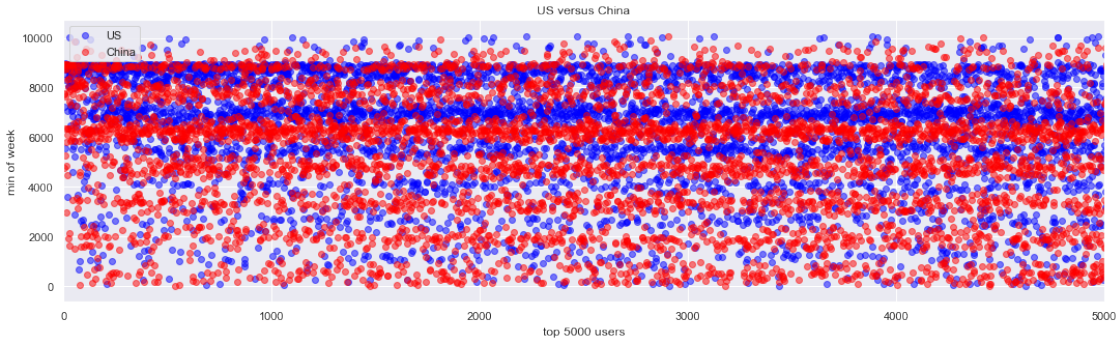
```
              precision    recall  f1-score   support

           2       0.81      0.99      0.89     10000
           8       0.99      0.76      0.86     10000

   micro avg       0.88      0.88      0.88     20000
   macro avg       0.90      0.88      0.88     20000
weighted avg       0.90      0.88      0.88     20000

[[9950   50]
 [2353 7647]]
[0.878125 0.88225  0.882625 0.874    0.878875 0.8705   0.88     0.877125
 0.876625 0.878375] 0.8778499999999999 0.0034504528688274953
```

In [20]:
```
# eda
# what do time stamps look like
plt.figure(figsize=(18,5))
plt.scatter(y=df_us.minute_of_week, x=list(range(50000)), label='US', color='blue', al
plt.scatter(y=df_uk.minute_of_week, x=list(range(50000)), label='UK', color='red', alp
plt.xlabel('top 5000 users')
plt.ylabel('min of week')
plt.title('US versus UK')
plt.legend()
plt.xlim(0, 5000)

plt.show()
```

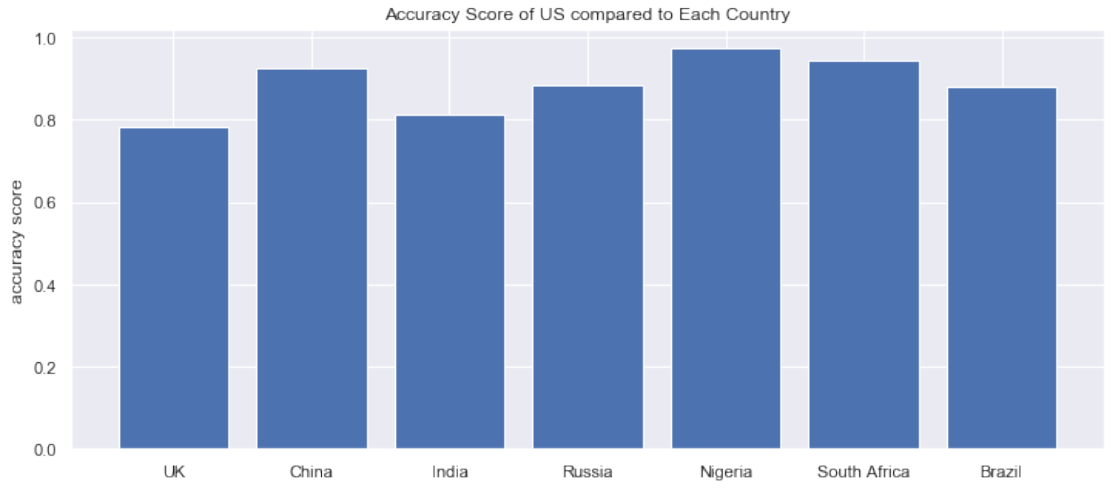


In [21]:
```
# eda
# what do time stamps look like
plt.figure(figsize=(18,5))
plt.scatter(y=df_us.minute_of_week, x=list(range(50000)), label='US', color='blue', al
plt.scatter(y=df_chn.minute_of_week, x=list(range(49629)), label='China', color='red'
plt.xlabel('top 5000 users')
plt.ylabel('min of week')
```

```
plt.title('US versus China')
plt.legend()
plt.xlim(0, 5000)

plt.show()
```



In [22]:
```
# how different are time signals from US based on predictive models
# get scores

score_list = [acc_us_uk, acc_us_chn, acc_us_ind, acc_us_rus, acc_us_ng, acc_us_sa, acc

x = np.arange(7)
plt.figure(figsize=(12,5))
plt.bar(x, score_list)
plt.xticks(x, ('UK', 'China', 'India', 'Russia', 'Nigeria', 'South Africa', 'Brazil'))
plt.ylabel('accuracy score')
plt.title('Accuracy Score of US compared to Each Country')
plt.show()
print('US most similar to UK')
print('most different from Nigeria')
print('model has more difficult time distinguishing between UK and US then US and Nige
```



```
US most similar to UK
most different from Nigeria
model has more difficult time distinguishing between UK and US then US and Nigeria
```

In [ ]: