In [69]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import timedelta
from datetime import datetime as dt
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.cluster import KMeans,MeanShift, estimate_bandwidth
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.ensemble import ExtraTreesClassifier
%matplotlib inline
```

In [2]:
```
# data set include 3 csv files
d1= "activity_log.csv"
d2= "enrollment_list.csv"
d3= "train_label.csv"
```

In [3]:
```
df_1 = pd.read_csv(d1)
df_1.head()
```

Out[3]:

	enrollment_id	time	event
0	1	2014-05-31T12:43:20	navigate
1	1	2014-05-31T12:43:50	navigate
2	1	2014-05-31T12:44:10	access
3	1	2014-05-31T12:56:32	navigate
4	1	2014-05-31T12:56:53	access

In [4]:
```
df_2= pd.read_csv(d2)
df_2.head()
```

Out[4]:

	enrollment_id	user_id	course_id
0	1	Mv7P1v8fRDifebDlgli2Vl1Z	ev4oSPQOjeL5QAf5oF72ooYu
1	2	xR26xfShRqzZY2EcRUj1yUtH	kxtZS4d61l2cEp0BZ3e6HzOH
2	3	7Ls723Xz49U2sCOqIKKtFMiW	TRrZ9gGs6MrmfBbpM1B9hzbn
3	4	RqpfcyRE7XYeyNurvR8s5i9T	nknoexvjeVdLxiDT0VODT9CV
4	5	1IxywohA3Ug7ok0p2MEDnFHY	djBWhu0JoDsrQ2a6Kzg6B4E2

In [5]:
```
# category of event
list(set(df_1.event.values))
```

Out[5]:
```
['problem', 'access', 'video', 'discussion', 'page_close', 'wiki', 'navigate']
```

In [6]:
```
#format the time
df_1['time'] = pd.to_datetime(df_1['time'])
df_1.dtypes
```

Out[6]:
```
enrollment_id             int64
time              datetime64[ns]
event                     object
dtype: object
```

In [7]:
```
df_1_1= df_1.groupby(['enrollment_id','event'])['time'].agg(['min','max']).reset_index()
df_1_1.head()
```

Out[7]:

	enrollment_id	event	min	max
0	1	access	2014-05-31 12:44:10	2014-06-01 14:00:30
1	1	navigate	2014-05-31 12:43:20	2014-06-01 14:00:21
2	1	problem	2014-05-31 12:59:40	2014-06-01 13:40:31
3	2	access	2014-06-04 02:58:59	2014-06-04 03:16:11
4	2	discussion	2014-06-04 02:57:03	2014-06-04 02:58:37

In [10]:
```
df_1_1['day_duration']=(df_1_1['max'] - df_1_1['min']).dt.days
df_1_1['mins_duration']=(df_1_1['max'] - df_1_1['min']).dt.total_seconds()/60

df_1_1.head()
```

Out[10]:

	enrollment_id	event	min	max	day_duration	mins_duration
0	1	access	2014-05-31 12:44:10	2014-06-01 14:00:30	1	1516.333333
1	1	navigate	2014-05-31 12:43:20	2014-06-01 14:00:21	1	1517.016667
2	1	problem	2014-05-31 12:59:40	2014-06-01 13:40:31	1	1480.850000
3	2	access	2014-06-04 02:58:59	2014-06-04 03:16:11	0	17.200000
4	2	discussion	2014-06-04 02:57:03	2014-06-04 02:58:37	0	1.566667

In [11]:
```
df_1_1.drop(['min', 'max'],axis= 1, inplace=True)
df_1_1.head()
```

Out[11]:

	enrollment_id	event	day_duration	mins_duration
0	1	access	1	1516.333333
1	1	navigate	1	1517.016667
2	1	problem	1	1480.850000
3	2	access	0	17.200000
4	2	discussion	0	1.566667

In [12]:
```
df_final = df_1_1.pivot(index = 'enrollment_id',columns = 'event').fillna(0)
df_final.columns=['_'.join(col).strip() for col in df_final.columns.values]
df_final.reset_index(inplace=True)
df_final.head()
```

Out[12]:

	enrollment_id	day_duration_access	day_duration_discussion	day_duration_navigate	day_duration_page_close	day_duration_problem
0	1	1.0	0.0	1.0	0.0	1.0
1	2	0.0	0.0	0.0	0.0	0.0
2	3	3.0	0.0	3.0	3.0	0.0
3	4	0.0	0.0	0.0	0.0	0.0
4	5	0.0	0.0	0.0	0.0	0.0

In [13]:
```
df_final_1= df_1.groupby(['enrollment_id','event'])['time'].count().unstack().fillna(0).reset_index()
df_final_1.head()
```

Out[13]:

event	enrollment_id	access	discussion	navigate	page_close	problem	video	wiki
0	1	27.0	0.0	8.0	0.0	79.0	0.0	0.0
1	2	7.0	4.0	3.0	4.0	0.0	3.0	1.0
2	3	34.0	1.0	6.0	7.0	0.0	9.0	0.0
3	4	54.0	4.0	6.0	46.0	21.0	31.0	1.0
4	5	0.0	0.0	2.0	0.0	0.0	0.0	0.0

In [14]:
```
df_final = df_final.merge(df_final_1,on='enrollment_id').reset_index()
df_final.head()
```

Out[14]:

	index	enrollment_id	day_duration_access	day_duration_discussion	day_duration_navigate	day_duration_page_close	day_duration_p
0	0	1	1.0	0.0	1.0	0.0	
1	1	2	0.0	0.0	0.0	0.0	
2	2	3	3.0	0.0	3.0	3.0	
3	3	4	0.0	0.0	0.0	0.0	
4	4	5	0.0	0.0	0.0	0.0	

5 rows × 23 columns

In [16]:
```
df_final.drop('index',axis=1,inplace=True)
df_final.head()
```

Out[16]:

	enrollment_id	day_duration_access	day_duration_discussion	day_duration_navigate	day_duration_page_close	day_duration_problem
0	1	1.0	0.0	1.0	0.0	1.0
1	2	0.0	0.0	0.0	0.0	0.0
2	3	3.0	0.0	3.0	3.0	0.0
3	4	0.0	0.0	0.0	0.0	0.0
4	5	0.0	0.0	0.0	0.0	0.0

5 rows × 22 columns

In [17]:
```
df_2= pd.read_csv(d2)
df_2.head()
```

Out[17]:

	enrollment_id	user_id	course_id
0	1	Mv7P1v8fRDifebDlgIi2Vl1Z	ev4oSPQOjeL5QAf5oF72ooYu
1	2	xR26xfShRqzZY2EcRUj1yUtH	kxtZS4d61l2cEp0BZ3e6HzOH
2	3	7Ls723Xz49U2sCOqIKKtFMiW	TRrZ9gGs6MrmfBbpM1B9hzbn
3	4	RqpfcyRE7XYeyNurvR8s5i9T	nknoexvjeVdLxiDT0VODT9CV
4	5	1IxywohA3Ug7ok0p2MEDnFHY	djBWhu0JoDsrQ2a6Kzg6B4E2

In [18]:
```
df_2_g=df_2.groupby('user_id')['course_id'].count().reset_index(name = 'count').sort_values(by ='count',ascen
ding = False)
len(df_2_g['count']>1)/df_2.shape[0]
```

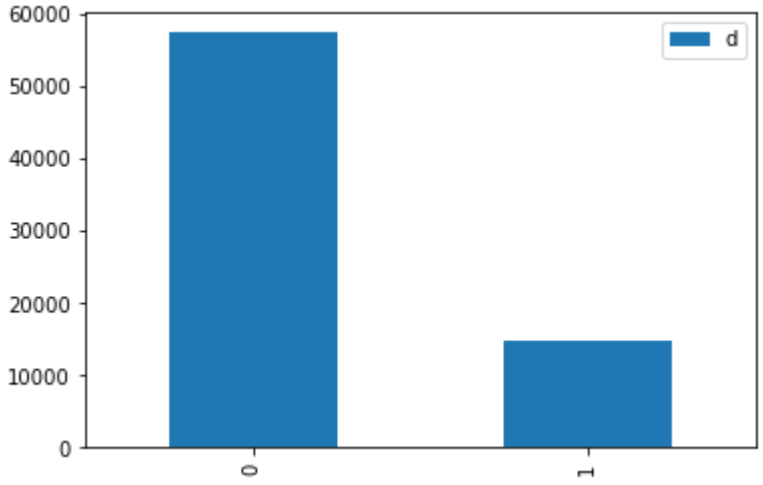
Out[18]: 0.6569162615519902

In [19]:
```
df_3= pd.read_csv(d3)
df_3.head()
```

Out[19]:

	enrollment_id	dropout_prob
0	1	1
1	2	1
2	3	0
3	4	1
4	5	1

In [20]:
```
# see the dropout distribution
df_3['dropout_prob'].value_counts().rename_axis('dropout_prob').reset_index(name='count')['count'].plot(kind
='bar')
plt.legend('dropout')
plt.show()
```



In [23]:
```
def statbyfeature(data,features):
    user_stat =dict()
    user = data['user_id'].nunique()
    for feature in features:
        seg = data[['user_id',feature]]
        seg_res = seg.groupby('user_id')[feature].nunique().reset_index(name = feature+"_nums")
        user_stat[feature] = seg_res
    res = user_stat[features[0]]
    for feature in features[1:]:
        res = pd.merge(res,user_stat[feature],on= 'user_id')
    return res
```

In [27]:
```
# merge  "enrollment_list.csv" and   "train_label.csv"
df = pd.merge(df_2,df_3, on = 'enrollment_id')
df.head()
```

Out[27]:

	enrollment_id	user_id	course_id	dropout_prob
0	1	Mv7P1v8fRDifebDlgIi2Vl1Z	ev4oSPQOjeL5QAf5oF72ooYu	1
1	2	xR26xfShRqzZY2EcRUj1yUtH	kxtZS4d61l2cEp0BZ3e6HzOH	1
2	3	7Ls723Xz49U2sCOqIKKtFMiW	TRrZ9gGs6MrmfBbpM1B9hzbn	0
3	4	RqpfcyRE7XYeyNurvR8s5i9T	nknoexvjeVdLxiDT0VODT9CV	1
4	5	1IxywohA3Ug7ok0p2MEDnFHY	djBWhu0JoDsrQ2a6Kzg6B4E2	1

In [28]:
```
features = ['user_id','course_id']

df1 = statbyfeature(df,['course_id'])
df = pd.merge(df,df1,on='user_id')
df = df.merge(df.groupby('course_id')['user_id'].nunique().reset_index(name = 'user_id_nums'),on ='course_id'
)


df.head()
```

Out[28]:

	enrollment_id	user_id	course_id	dropout_prob	course_id_nums	user_id_nums
0	1	Mv7P1v8fRDifebDlgIi2Vl1Z	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819
1	10	j5ZHV45UBuKGT6sh9xpKebmX	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819
2	19	3drIIw6edZ7wcz6ttIW1rAdx	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819
3	24	BnlQkllgZzIwM9pQFXL5b0tx	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819
4	36756	1tXfa1rxy8JDbfsS7YL99taf	ev4oSPQOjeL5QAf5oF72ooYu	1	12	2819

In [29]:
```
def countByFeature_risk(df,f_count):
    dfs = df
    for f in f_count:
        tot = df.groupby(f)['dropout_prob'].sum()
        cnt = df.groupby(f)['dropout_prob'].count()
        tmpt =pd.DataFrame(tot/cnt).reset_index()
        tmpt.columns =[f,str(f+"_risk")]
        dfs = pd.merge(dfs,tmpt,on = f)
        return dfs
```

In [30]:
```
features = ['user_id','course_id']

dfff= countByFeature_risk(df,features)
features = ['course_id']
dfff2= countByFeature_risk(df[['enrollment_id','course_id','dropout_prob']],features)
dfff2.drop(['dropout_prob','course_id'],axis=1,inplace=True)
dfff2.head()
```

Out[30]:

	enrollment_id	course_id_risk
0	1	0.826534
1	10	0.826534
2	19	0.826534
3	24	0.826534
4	36756	0.826534

In [32]:
```
# merge the two different features generated dataset
df_final_2 = dfff.merge(dfff2,on='enrollment_id')
```

In [33]:
```
df_final_2.head()
```

Out[33]:

	enrollment_id	user_id	course_id	dropout_prob	course_id_nums	user_id_nums	user_id_risk
0	1	Mv7P1v8fRDifebDlgli2Vl1Z	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819	1.000000
1	10	j5ZHV45UBuKGT6sh9xpKebmX	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819	1.000000
2	19	3drIIw6edZ7wcz6ttIW1rAdx	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819	1.000000
3	24	BnlQkllgZzIwM9pQFXL5b0tx	ev4oSPQOjeL5QAf5oF72ooYu	1	1	2819	1.000000
4	36756	1tXfa1rxy8JDbfsS7YL99taf	ev4oSPQOjeL5QAf5oF72ooYu	1	12	2819	0.666667

In [34]:
```
# merge the all feature generated dataset
df_f = df_final.merge(df_final_2,on = 'enrollment_id')

df_f.head()
```

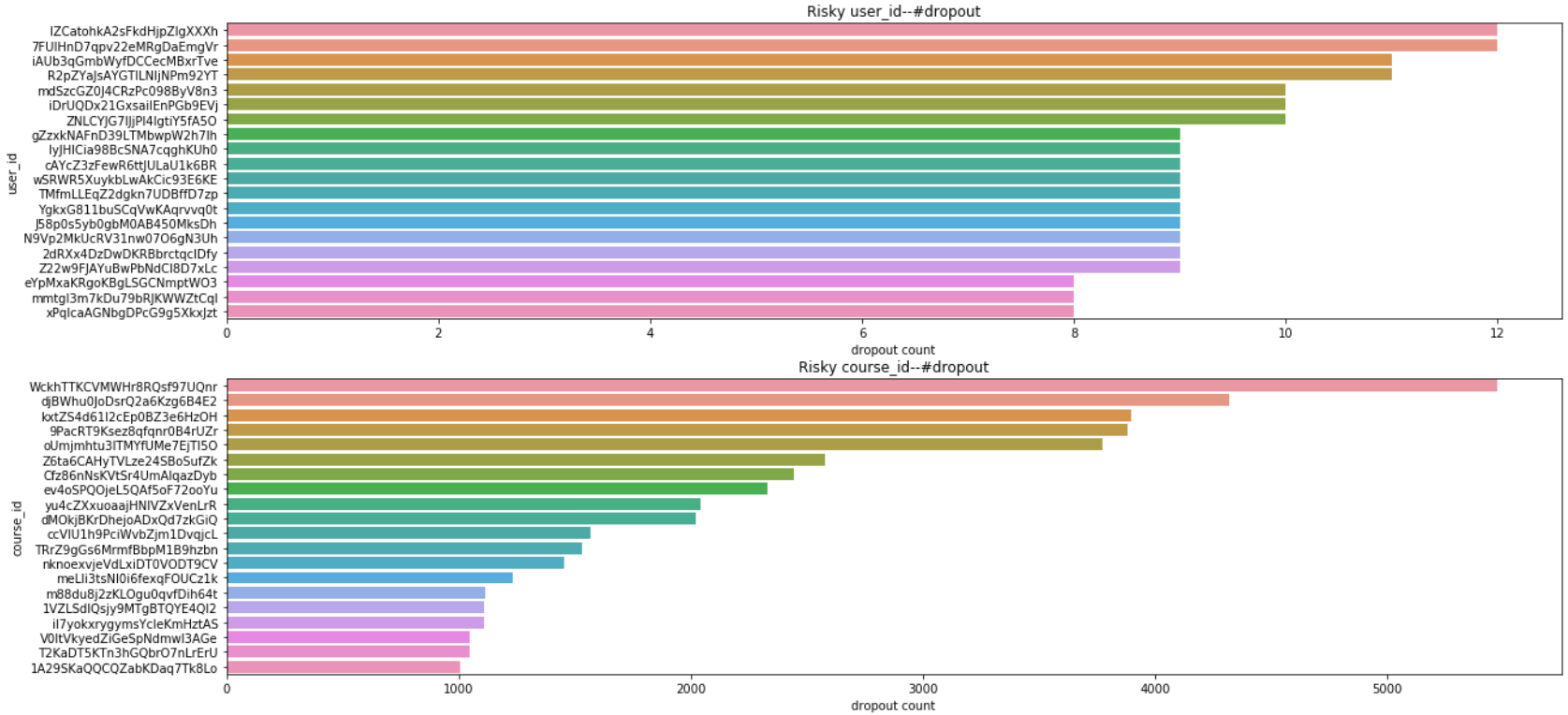
Out[34]:

	enrollment_id	day_duration_access	day_duration_discussion	day_duration_navigate	day_duration_page_close	day_duration_problem
0	1	1.0	0.0	1.0	0.0	1.0
1	2	0.0	0.0	0.0	0.0	0.0
2	3	3.0	0.0	3.0	3.0	0.0
3	4	0.0	0.0	0.0	0.0	0.0
4	5	0.0	0.0	0.0	0.0	0.0

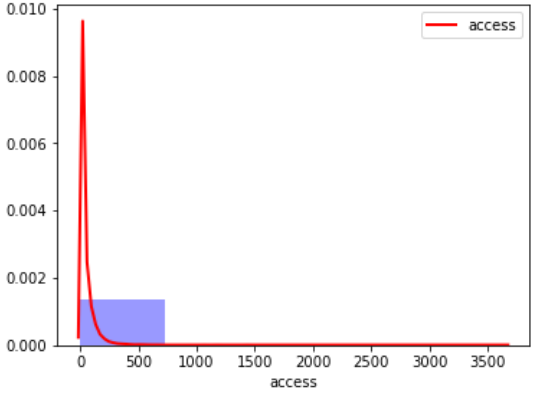
5 rows × 29 columns

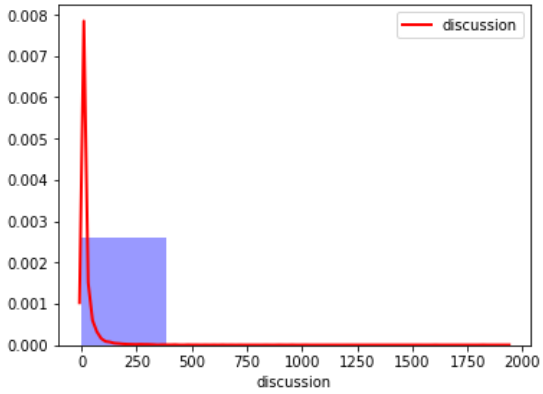
Exploratory Data Analysis

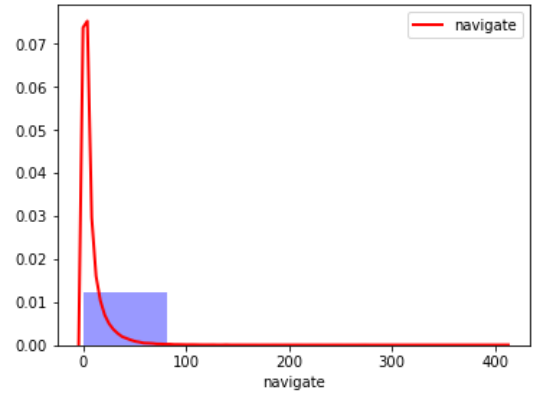
In [91]:
```
# users  and course risk analysis
features =['user_id','course_id']
index =0
plt.figure(figsize=(20,10))

for f in features:
    index +=1
    data = df.groupby(f)['dropout_prob'].sum().sort_values(ascending=False).reset_index(name = 'dropout count')[:20]
    plt.subplot(2,1,index)
    plt.title("Risky "+str(f)+"--#dropout")
    plt.xlabel(fea)
    sns.barplot(y=f,x ='dropout count',data =data)
plt.show()
```

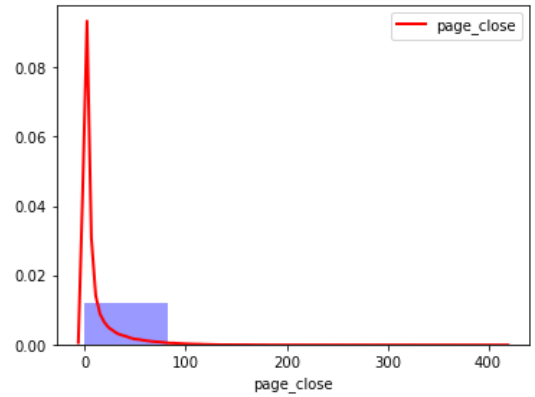


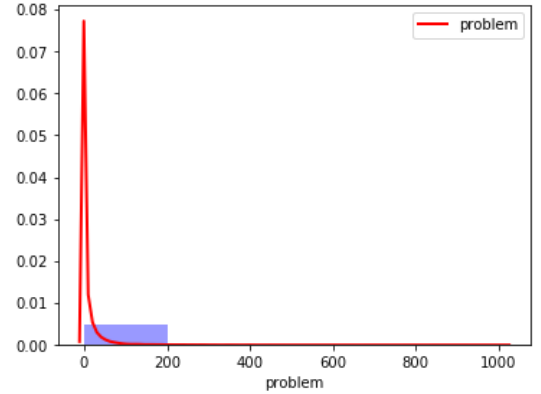
In [111]:
```
# event histogram
features = ['access', 'discussion', 'navigate', 'page_close', 'problem', 'video','wiki']
index = 0
plt.figure(figsize= (20,20))
data =df_f
for f in features:
    index += 1
    plt.subplot(4,3,index)
    plt.xlabel(f)
    sns.distplot(data[f],kde = True, bins =5,color= 'b',kde_kws = {"color":'r',"lw":2,"label":f})
```

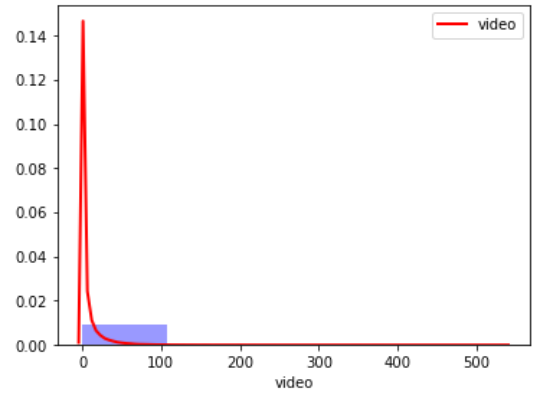

access
0.010
0.008
0.006
0.004
0.002
0.000
0
500
1000
1500
2000
2500
3000
3500
access

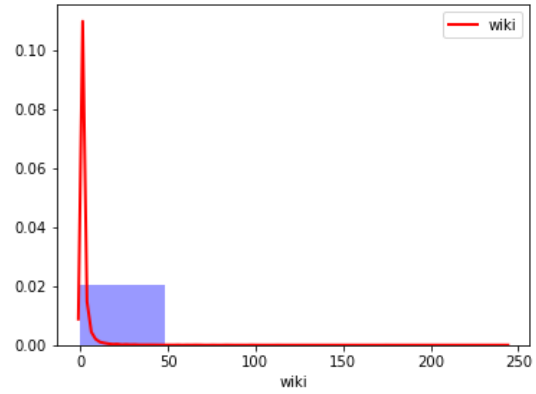

discussion
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
250
500
750
1000
1250
1500
1750
2000
discussion


navigate
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
navigate

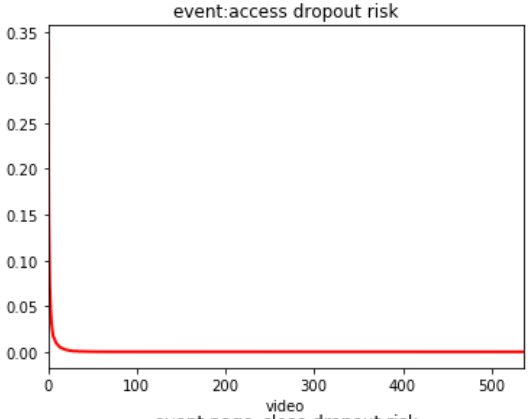

page_close
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
page_close

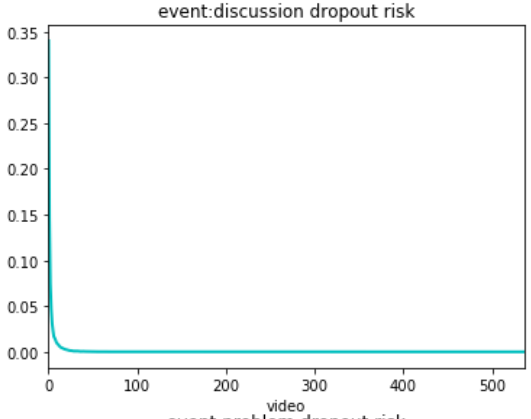

problem
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
200
400
600
800
1000
problem

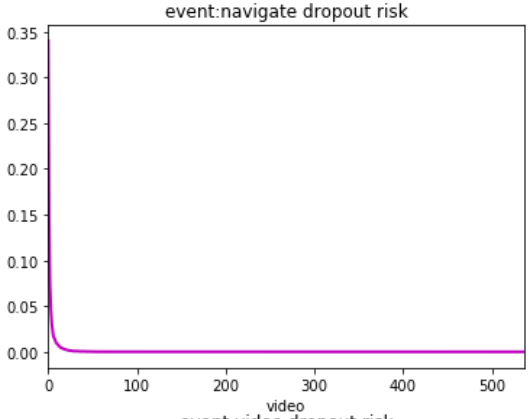

video
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
500
video

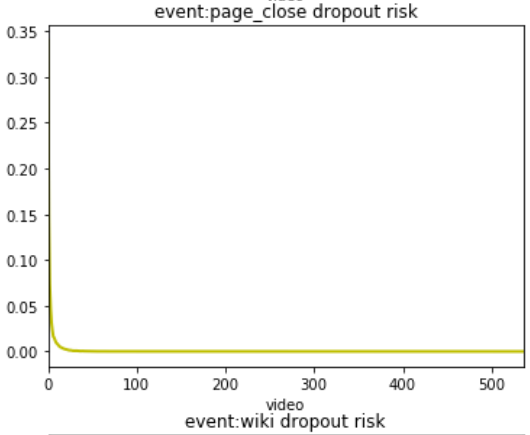

wiki
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
wiki

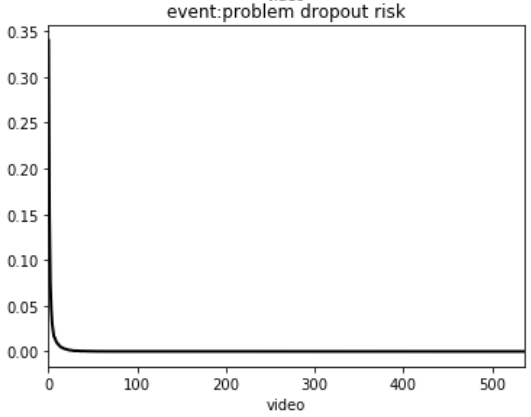
In [138]:
```
# event risk line chart
features = ['access', 'discussion', 'navigate', 'page_close', 'problem', 'video','wiki']
index = 0
plt.figure(figsize= (20,20))
data =df_f
colors = ['','r','c','m','y','k','g','b']
for _,f in enumerate(features):
    index += 1
    plt.subplot(4,3,index)

    d = data.groupby('video')['dropout_prob'].sum()/(data.shape[0])
    plt.title('event:'+str(f)+ ' dropout risk')
    plt.xlabel(""+str(f)+" count")
    d.plot(color= colors[index], linewidth=2)
```

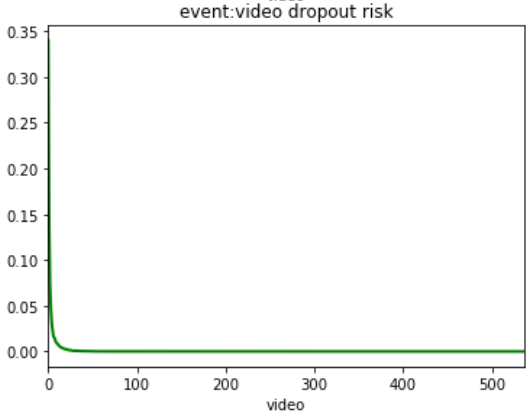


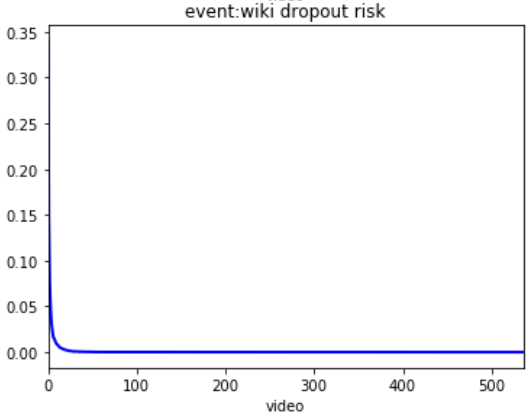












Train Model and evaluate

In [36]:
```
x = df_f.drop(['enrollment_id','user_id','course_id','dropout_prob'],axis=1)
y = df_f['dropout_prob']
x.shape
```

Out[36]:
```
(72325, 25)
```

In [66]:
```
# split training and test dataset
x_train,x_test ,y_train, y_test = train_test_split(StandardScaler().fit_transform(x),y,test_size =.33,random_
state = 42)
```

In [147]:
```
#model build and performance evaluate
def pred_model(x_train,x_test ,y_train, y_test):
    Lg =  LogisticRegression(max_iter = 2000)
    eval_set = [(x_test, y_test)]
    Xgboost= XGBClassifier(eval_metric=["error", "logloss"], eval_set=eval_set,verbose=True,
                           objective='binary:logistic', n_estimators=100 )
    gnb = GaussianNB()
    rdf = RandomForestClassifier(max_depth=10, random_state=0)

    models_name = ["LogisticRegression","XGBoost","GaussianNB","RandomForest"]
    models = [Lg,Xgboost,gnb,rdf]
    for index,model in enumerate(models):
        print(f"*********{models_name[index]}********")
        model.fit(x_train,y_train)
        fpr,tpr,  _ = metrics.roc_curve(y_test,model.predict_proba(x_test)[:,1])
        auc = metrics.auc(fpr,tpr)
        y_pred = model.predict(x_test)
        y_pred_train = model.predict(x_train)
        f1_test = f1_score(y_test, y_pred,average='micro')
        f1_train = f1_score(y_train, y_pred_train,average='micro')

        print(f'AUC score:   {auc:4.3f}')
        print(f"train score: {model.score(x_train,y_train):4.3f}")
        print(f"test score:  {model.score(x_test,y_test):4.3f}")

        print("F1 score for train dataset: ",f"{f1_train:4.3f}")
        print("F1 score for test dataset:  ",f"{f1_test:4.3f}")
        print()
pred_model(x_train,x_test ,y_train, y_test)
```

```
*********LogisticRegression********
AUC score:   0.991
train score: 0.958
test score:  0.957
F1 score for train dataset:  0.958
F1 score for test dataset:   0.957

*********XGBoost********
AUC score:   0.992
train score: 0.985
test score:  0.960
F1 score for train dataset:  0.985
F1 score for test dataset:   0.960

*********GaussianNB********
AUC score:   0.943
train score: 0.873
test score:  0.871
F1 score for train dataset:  0.873
F1 score for test dataset:   0.871

*********RandomForest********
AUC score:   0.992
train score: 0.970
test score:  0.961
F1 score for train dataset:  0.970
F1 score for test dataset:   0.961
```

Feature importance analysis

In [71]:
```
features = x.columns
features
```

Out[71]:
```
Index(['day_duration_access', 'day_duration_discussion',
       'day_duration_navigate', 'day_duration_page_close',
       'day_duration_problem', 'day_duration_video', 'day_duration_wiki',
       'mins_duration_access', 'mins_duration_discussion',
       'mins_duration_navigate', 'mins_duration_page_close',
       'mins_duration_problem', 'mins_duration_video', 'mins_duration_wiki',
       'access', 'discussion', 'navigate', 'page_close', 'problem', 'video',
       'wiki', 'course_id_nums', 'user_id_nums', 'user_id_risk',
       'course_id_risk'],
      dtype='object')
```

In [81]:
```
# Build a forest and compute the feature importances

forest = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)

forest.fit(x, y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1][:10]

# Print the feature ranking
print("Feature ranking:")

for f in range(10):

    print("%d. feature %d: %s, (%f)" % (f + 1, indices[f],features[indices[f]], importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.barh(range(10), importances[indices][::-1],
       color="r", yerr=std[indices], align="center")

#plt.xticks(range(10), features[indices],rotation = 90)
plt.yticks(range(10), features[indices])
#plt.xlim([-1, x.shape[1]])
plt.show()
```

```
Feature ranking:
1. feature 23: user_id_risk, (0.529231)
2. feature 7: mins_duration_access, (0.044646)
3. feature 4: day_duration_problem, (0.038590)
4. feature 11: mins_duration_problem, (0.036919)
5. feature 10: mins_duration_page_close, (0.036340)
6. feature 0: day_duration_access, (0.031480)
7. feature 9: mins_duration_navigate, (0.031035)
8. feature 3: day_duration_page_close, (0.030489)
9. feature 2: day_duration_navigate, (0.026075)
10. feature 12: mins_duration_video, (0.023930)
```