In [122]:
```
# The following code references from
# https://github.com/yhat/DataGotham2013/tree/master/notebooks
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import pylab as pl

# import pipline files:
import read
import explore
import clean
import gen_features
import classifier
import evaluate

#Global Variable:
explore.DEP_VAR ='SeriousDlqin2yrs'
explore.LEAD_VAR = 2
gen_features.DEP_VAR ='SeriousDlqin2yrs'
classifier.DEP_VAR ='SeriousDlqin2yrs'
```

In [123]:
```
# Read/load data
df = read.read("credit-data.csv")
# Adjust data type -percentage
df.RevolvingUtilizationOfUnsecuredLines = df.RevolvingUtilizationOfUns
ecuredLines*100
df.DebtRatio = df.DebtRatio*100
df.head()
```

Out[123]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode	N 5
0	1	1	76.612661	45	60644	2
1	2	0	95.715102	40	60637	0
2	3	0	65.818014	38	60601	1
3	4	0	23.380978	30	60601	0
4	5	0	90.723940	49	60625	1

In [124]:
```
d = explore.explore(df)
summary = d["summary"]
features = d["features"]
```

In [125]:
```
features
```

Out[125]:
```
['RevolvingUtilizationOfUnsecuredLines',
 'age',
 'zipcode',
 'NumberOfTime30-59DaysPastDueNotWorse',
 'DebtRatio',
 'MonthlyIncome',
 'NumberOfOpenCreditLinesAndLoans',
 'NumberOfTimes90DaysLate',
 'NumberRealEstateLoansOrLines',
 'NumberOfTime60-89DaysPastDueNotWorse',
 'NumberOfDependents']
```

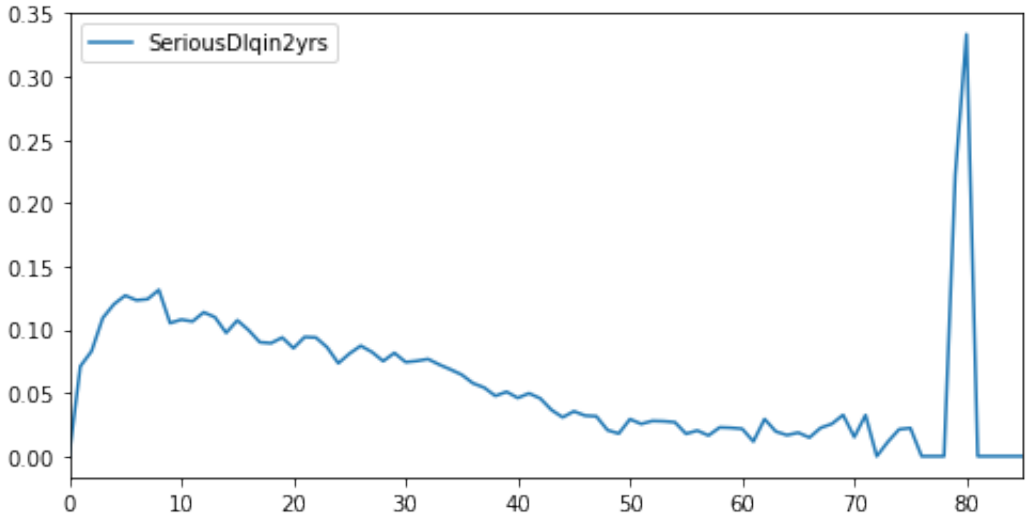
In [126]:
```
# Summary statistics for the whole dataset
summary
```

Out[126]:

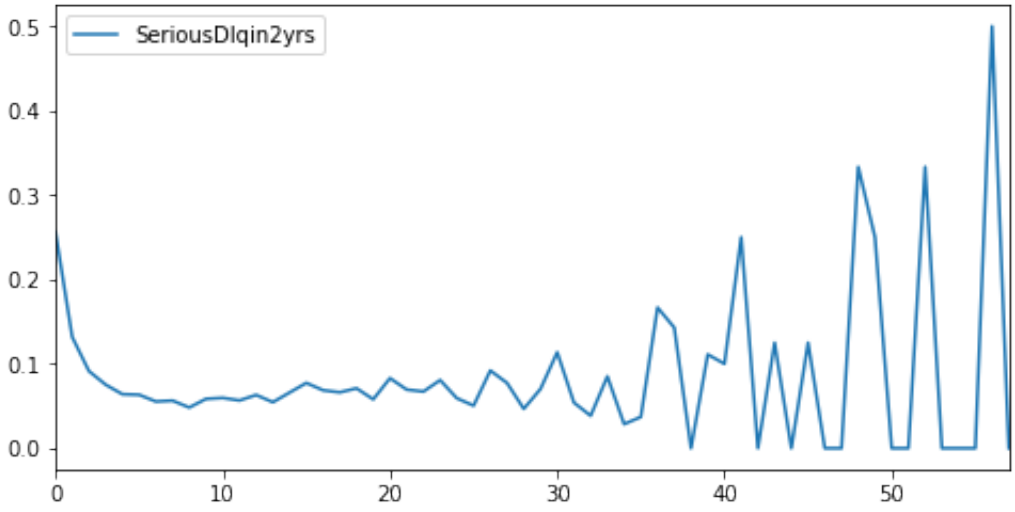
	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age
count	150000.000000	150000.000000	1.500000e+05	15000
mean	75000.500000	0.066840	6.048438e+02	52.295
std	43301.414527	0.249746	2.497554e+04	14.771
min	1.000000	0.000000	0.000000e+00	0.000
25%	37500.750000	0.000000	2.986744e+00	41.000
50%	75000.500000	0.000000	1.541807e+01	52.000
75%	112500.250000	0.000000	5.590462e+01	63.000
max	150000.000000	1.000000	5.070800e+06	109.00

In [127]:
```
# Plot distribution for variable 'age', 'NumberOfOpenCreditLinesAndLoa
ns'
# and 'NumberOfDependents'with respect to 'SeriousDlqin2yrs'
```

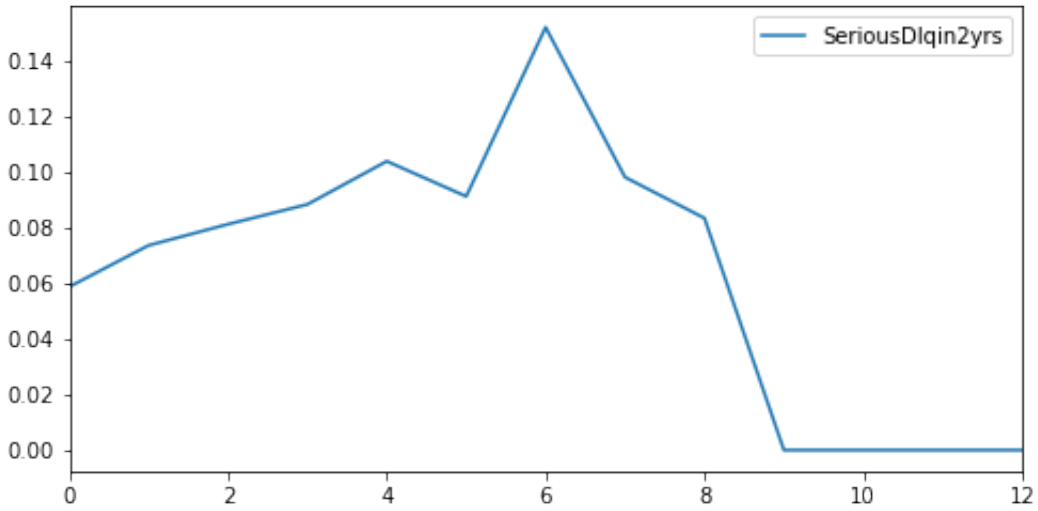
In [128]:
```
age_graph = explore.explore_var(df,'age','line')["graph"]
```



In [129]:
```
loan_graph = explore.explore_var(df,'NumberOfOpenCreditLinesAndLoans',
'line')["graph"]
```



In [130]:
```
dependent_graph = explore.explore_var(df,'NumberOfDependents','line')[
"graph"]
```



In [131]:
```
#check the null value
clean.check_missing_data(df)
```

Out[131]:

value	False	True
variable		
DebtRatio	150000	0
MonthlyIncome	120269	29731
NumberOfDependents	146076	3924
NumberOfOpenCreditLinesAndLoans	150000	0
NumberOfTime30-59DaysPastDueNotWorse	150000	0
NumberOfTime60-89DaysPastDueNotWorse	150000	0
NumberOfTimes90DaysLate	150000	0
NumberRealEstateLoansOrLines	150000	0
PersonID	150000	0
RevolvingUtilizationOfUnsecuredLines	150000	0
SeriousDlqin2yrs	150000	0
age	150000	0
zipcode	150000	0

In [132]:
```
#Check null values again after filling in missing values
df = clean.clean(df,'NumberOfDependents','zero')
df = clean.clean(df,'MonthlyIncome','mean')
clean.check_missing_data(df)
```

Out[132]:

value	False
variable	
DebtRatio	150000
MonthlyIncome	150000
NumberOfDependents	150000
NumberOfOpenCreditLinesAndLoans	150000
NumberOfTime30-59DaysPastDueNotWorse	150000
NumberOfTime60-89DaysPastDueNotWorse	150000
NumberOfTimes90DaysLate	150000
NumberRealEstateLoansOrLines	150000
PersonID	150000
RevolvingUtilizationOfUnsecuredLines	150000
SeriousDlqin2yrs	150000
age	150000
zipcode	150000

In [133]:
```
#Generate categorical bin boundary for selected variables
UnsecuredLines_bins = gen_features.generate_bins(df,'RevolvingUtilizat
ionOfUnsecuredLines',5)
DebtRatio_bins = gen_features.generate_bins(df,'DebtRatio',10)
income_bins = gen_features.generate_bins(df,'MonthlyIncome',500)
age_bins = gen_features.generate_bins(df,'age',5)

df = gen_features.build_category(df,'RevolvingUtilizationOfUnsecuredLi
nes',UnsecuredLines_bins)
df = gen_features.build_category(df,'DebtRatio',DebtRatio_bins)
df = gen_features.build_category(df,'MonthlyIncome',income_bins)
df = gen_features.build_category(df,'age',age_bins)
```

In [134]:
```
# Show df with newly added variables:
# "RevolvingUtilizationOfUnsecuredLines_bucket"
# "DebtRatio_bucket"
# "MonthlyIncome_bucket"
# "age_bucket"
df.describe()
```

Out[134]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age
count	150000.000000	150000.000000	1.500000e+05	15000
mean	75000.500000	0.066840	6.048438e+02	52.295
std	43301.414527	0.249746	2.497554e+04	14.771
min	1.000000	0.000000	0.000000e+00	0.000
25%	37500.750000	0.000000	2.986744e+00	41.000
50%	75000.500000	0.000000	1.541807e+01	52.000
75%	112500.250000	0.000000	5.590462e+01	63.000
max	150000.000000	1.000000	5.070800e+06	109.00

In [135]:
```
# Create dummy variables
DebtRatio_dummy = gen_features.create_dummy(df,'DebtRatio_bucket')
income_dummy = gen_features.create_dummy(df,'MonthlyIncome_bucket')
age_dummy = gen_features.create_dummy(df,'age_bucket')
```

In [136]:
```
# Concat the dummy variables to df
df_new = pd.concat([df, DebtRatio_dummy,income_dummy,age_dummy], axis=
1)
df_new.head()
```

Out[136]:

	PersonID	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode	N 59
0	1	1	76.612661	45	60644	2
1	2	0	95.715102	40	60637	0
2	3	0	65.818014	38	60601	1
3	4	0	23.380978	30	60601	0
4	5	0	90.723940	49	60625	1

5 rows × 40 columns

In [137]:
```
#Select model and train model
selected_features = [
 'RevolvingUtilizationOfUnsecuredLines',
 'DebtRatio_bucket_0',
 'DebtRatio_bucket_1',
 'DebtRatio_bucket_2',
 'DebtRatio_bucket_3',
 'DebtRatio_bucket_4',
 'DebtRatio_bucket_5',
 'DebtRatio_bucket_6',
 'DebtRatio_bucket_7',
 'DebtRatio_bucket_8',
 'MonthlyIncome_bucket_0',
 'MonthlyIncome_bucket_1',
 'MonthlyIncome_bucket_2',
 'MonthlyIncome_bucket_3',
 'MonthlyIncome_bucket_4',
 'MonthlyIncome_bucket_5',
 'MonthlyIncome_bucket_6',
 'MonthlyIncome_bucket_7',
 'age_bucket_0',
 'age_bucket_1',
 'age_bucket_2',
 'age_bucket_3',
 'age_bucket_4',
 'age_bucket_5']
```

In [138]:
```
(X_train, X_test, y_train, y_test) = classifier.form_train_test(df_new
,selected_features,0.3)
(yhat, probs) = classifier.classifier("KNN",X_train, X_test, y_train)
```

In [139]:
```
# Evaluate KNN classifier/model
(confusion_matrix, report) = evaluate.evaluate(y_test, yhat)
confusion_matrix
```

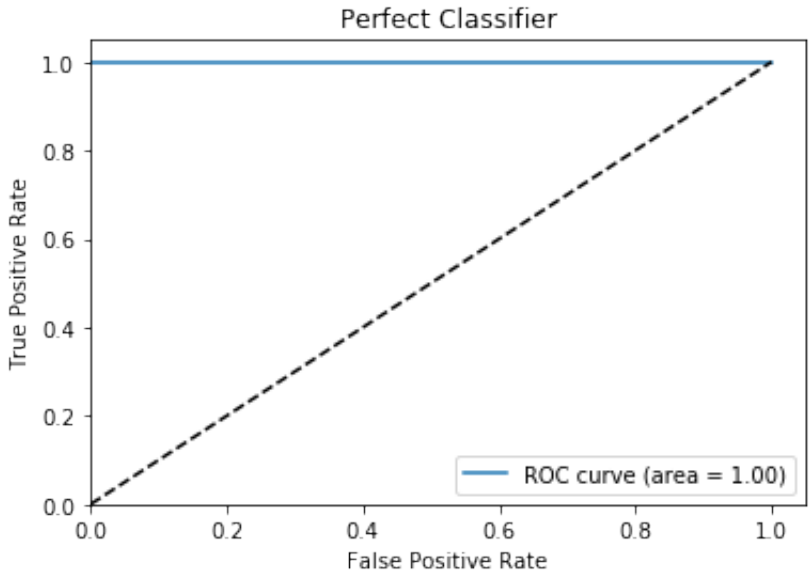
Out[139]:

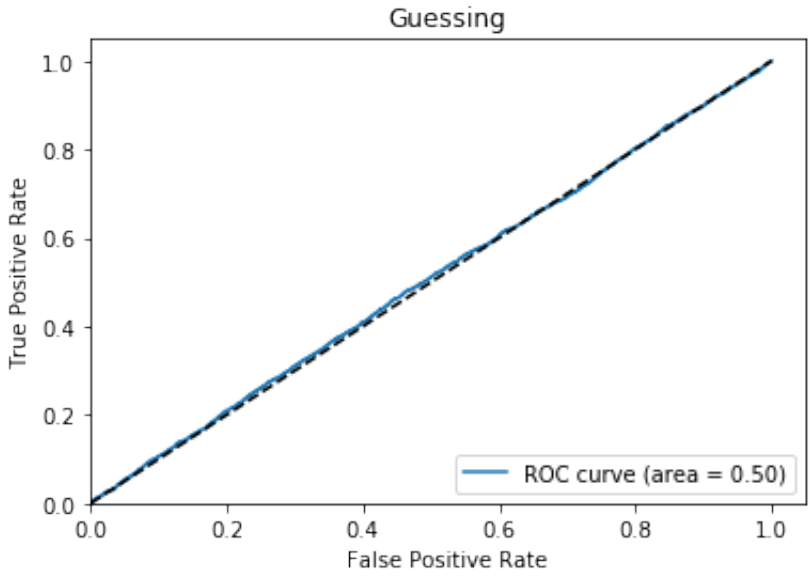
Predicted	0	1
Actual		
0	41316	2776
1	704	204

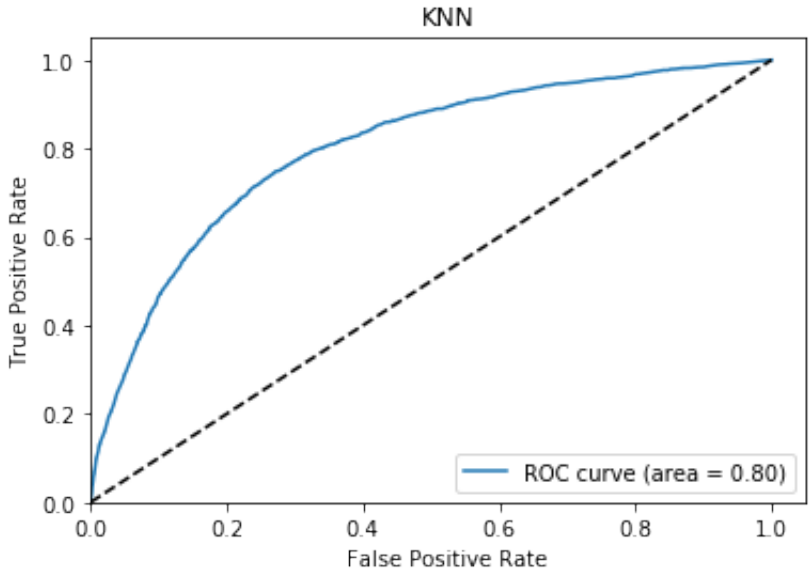
In [140]:
```
print(report)
```

```
             precision    recall  f1-score   support

          0       0.94      0.98      0.96     42020
          1       0.22      0.07      0.10      2980

avg / total       0.89      0.92      0.90     45000
```

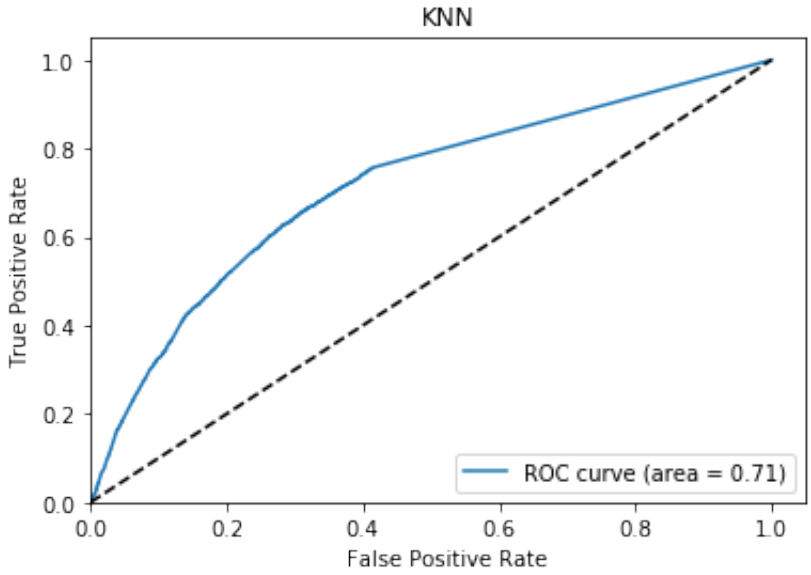
In [144]:
```
plot_roc("Perfect Classifier", y_test)
plot_roc("Guessing", np.random.uniform(0, 1, len(y_test)))
plot_roc("KNN", probs[::,1])
```

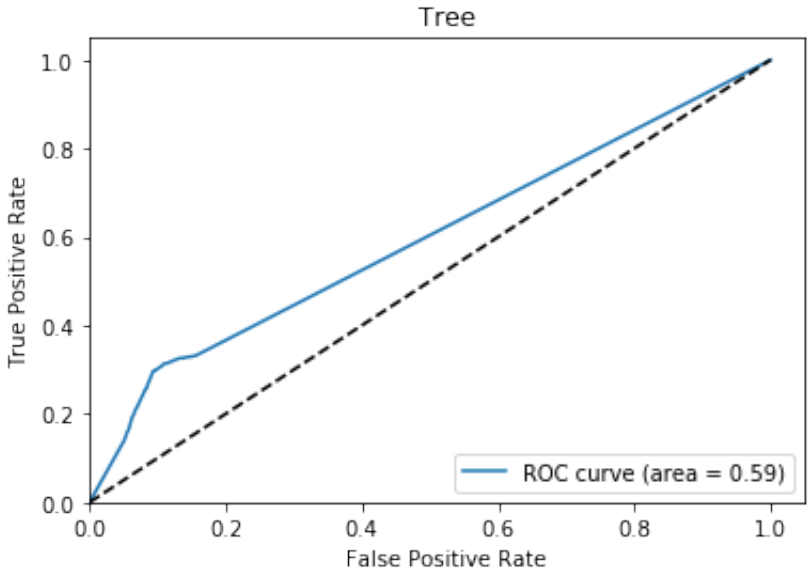


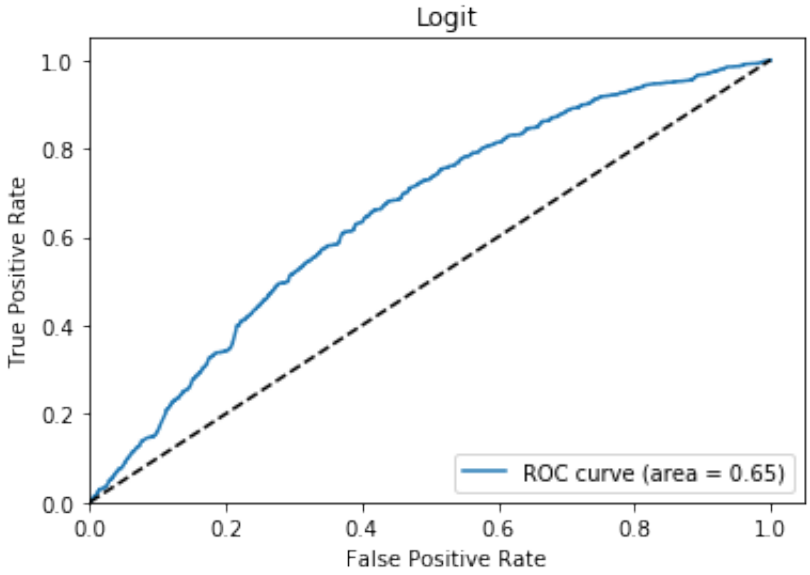


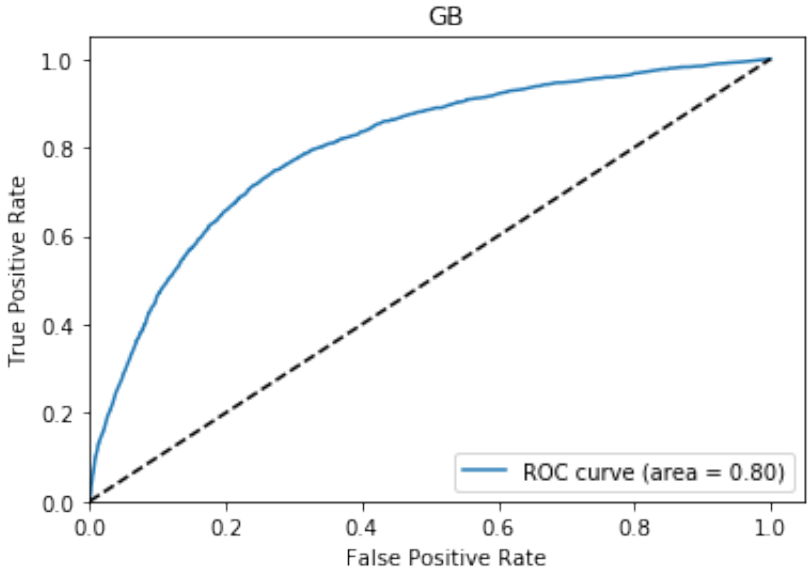


In [145]:
```
# Evaluate other classifier/model
models = ['KNN','Tree','Logit','GB']
for m in models:
    (yhat, probs) = classifier.classifier(m,X_train, X_test, y_train)
    (confusion_matrix, report) = evaluate.evaluate(y_test, yhat)
    plot_roc(m, probs[:,1])
```









In [146]:
```
# GB calssifier fit the data better
```