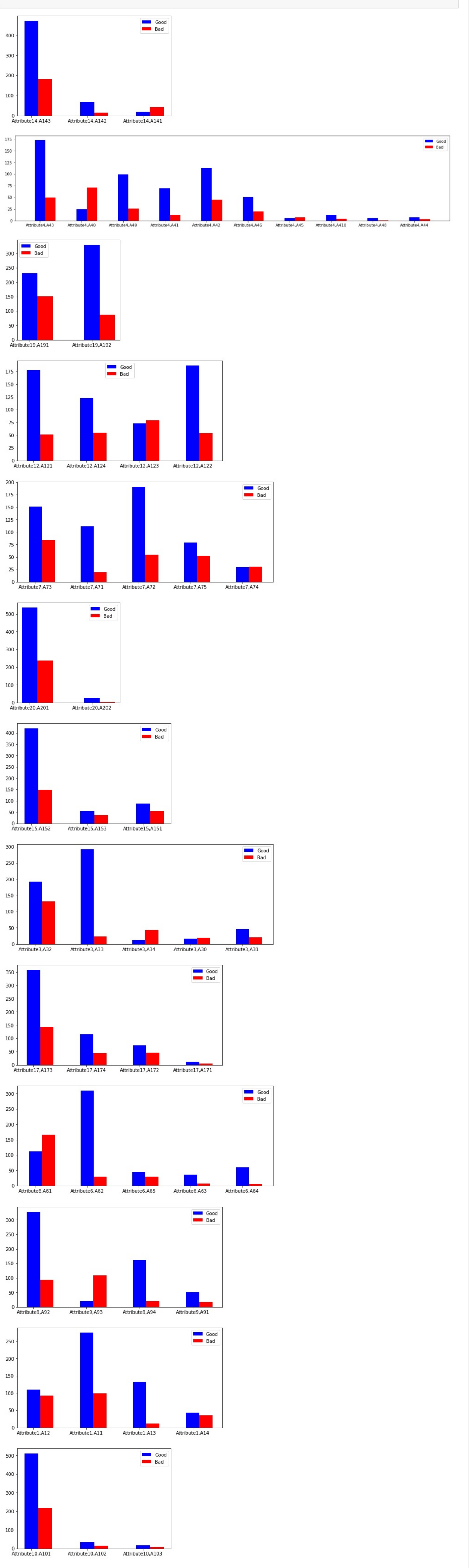
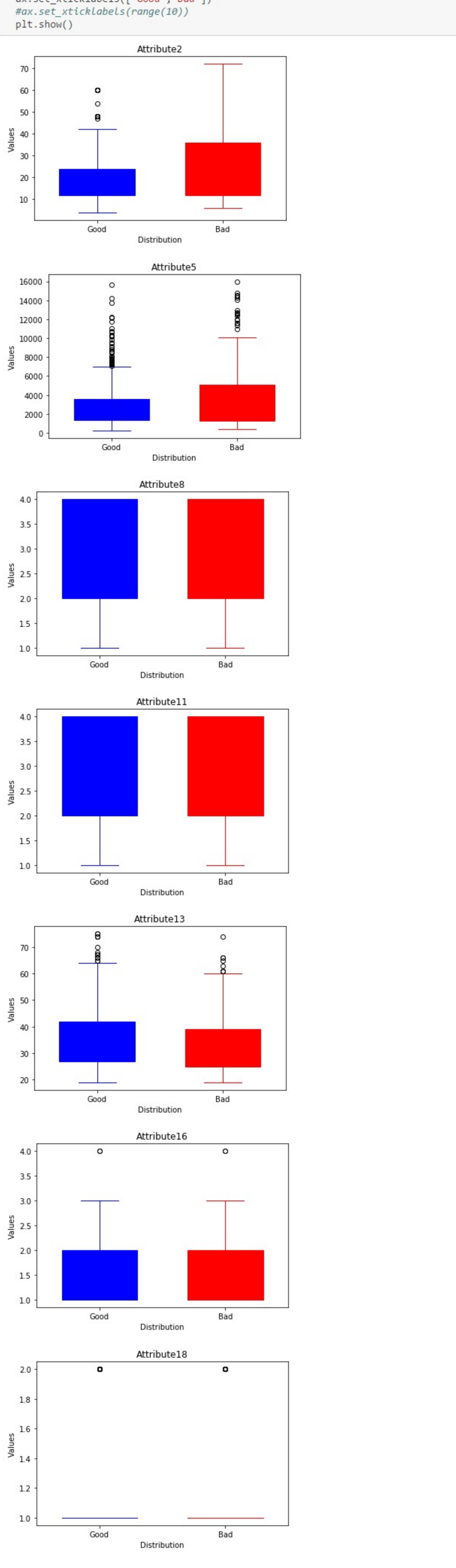
Categorical Visualization

In [1]:
```python
import pandas as pd

df = pd.read_csv('train.tsv', sep='\t')

from os import path
import matplotlib.pyplot as plt

cols_to_transform = list(set(df.columns)-set(df._get_numeric_data().columns))
new_cols=[]
for col in cols_to_transform:
    col += ','
    new_cols.append(col)

good = df[df["Label"]==1]
bad = df[df["Label"]==2]

cols_to_drop = df._get_numeric_data().columns
good_cat = good.drop( cols_to_drop, axis = 1 )
good_dict = good_cat.to_dict(orient='records')
bad_cat = bad.drop( cols_to_drop, axis = 1 )
bad_dict = bad_cat.to_dict(orient='records')

from sklearn.feature_extraction import DictVectorizer as DV

vectorizer = DV(separator=',',sparse=False)
goodvec = vectorizer.fit_transform(good_dict)
badvec = vectorizer.transform(bad_dict)
good2 = pd.DataFrame(goodvec,columns=vectorizer.get_feature_names())
bad2 = pd.DataFrame(badvec,columns=vectorizer.get_feature_names())
for single_col in new_cols:
    filter_col=[col for col in list(vectorizer.get_feature_names()) if col.startswith(single_col)]
    good3=good2[filter_col]
    bad3 = bad2[filter_col]
    from collections import Counter
    cnt = Counter()
    cnt2 = Counter()
    for index,row in good3.iterrows():
        for i in filter_col:
          if(row[i]==1.0):
            cnt[i]+=1
    for index,row in bad3.iterrows():
        for i in filter_col:
          if(row[i]==1.0):
            cnt2[i]+=1
    import numpy as np
    l = np.arange(len(cnt.keys()))
    width=0.25
    fig = plt.figure(figsize=(2 * len(cnt.keys()), 4))
    ax = fig.add_subplot(111)
    bars1 = ax.bar(l, cnt.values(), width,linewidth=2* len(cnt.keys()) * width,align='center',color='b')
    plt.xticks(l,cnt.keys())
    bars2 = ax.bar(l + width, cnt2.values(),width,linewidth =2 * len(cnt.keys()) * width, align='center',color='r')
    plt.xticks(l, cnt2.keys())
    ax.legend( (bars1[0], bars2[0]), ('Good', 'Bad') )
    plt.show()
```



Numerical Attributes Visualization(with box plots)

In [2]:
```python
def setBoxColors(bp):
    bp['boxes'][0].set( color='b',facecolor='b')
    bp['boxes'][1].set( color='r',facecolor='r' )

    bp['whiskers'][0].set( color='b')
    bp['whiskers'][1].set( color='b')
    bp['whiskers'][2].set( color='r')
    bp['whiskers'][3].set( color='r')

    bp['caps'][0].set( color='b')
    bp['caps'][1].set( color='b')
    bp['caps'][2].set( color='r')
    bp['caps'][3].set( color='r')


    bp['medians'][0].set( color='b')
    bp['medians'][1].set( color='r')

    bp['fliers'][0].set( color='b')
    bp['fliers'][1].set( color='r')

#ftiaxnoume ta dataframes mono me numerical values
cols_to_transform.append('Label')
cols_to_transform.append('Id')
box_good=good.drop(cols_to_transform, axis = 1)
box_bad=bad.drop(cols_to_transform, axis =1)
for x in box_good.columns:
  fig,ax = plt.subplots()
  data=[box_good[x],box_bad[x]]
  bp=ax.boxplot(data,positions = [1, 2], widths = 0.6,patch_artist=True)
  setBoxColors(bp)
  ax.set_title(x)
  ax.set_xlabel('Distribution')
  ax.set_ylabel('Values')
  ax.set_xticklabels(['Good','Bad'])
  #ax.set_xticklabels(range(10))
  plt.show()
```



Classification : SVM , RandomForest , GaussianNB

In [3]:
```python
import pandas as pd
import numpy as np

dataframe = pd.read_csv('train.tsv', sep='\t')
Y = dataframe['Label']

dataframe = dataframe.drop('Label',axis = 1)
dataframe = dataframe.drop('Id',axis = 1)

dataX = pd.get_dummies(dataframe)
X = dataX.as_matrix()
```

In [4]:
```python
from scipy import interp
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.model_selection import ShuffleSplit
from sklearn.metrics import accuracy_score

## SVM

k = 10
kf = ShuffleSplit(n_splits=k, test_size=0.1, random_state=0)
acc = 0

for train_index, test_index in kf.split(dataX):

    parameters = {'C':[1, 10]}
    svr = svm.LinearSVC()
    clf = GridSearchCV(svr, parameters)

    clf = clf.fit(X[train_index], Y[train_index])
    yPred = clf.predict(X[test_index])

    # Calculating accuracy
    acc += accuracy_score(yPred, Y[test_index])


em = np.zeros((1,3)) #storing mean accuracy for EvaluationMetric.csv

em[0,0] = acc/k
print ( "Average Accuracy of SVC_linear : " + str(acc/k) )
```

Average Accuracy of SVC_linear : 0.6675

In [5]:
```python
from sklearn.ensemble import RandomForestClassifier

kf = ShuffleSplit(n_splits=k, test_size=0.1, random_state=0)
acc = 0

for train_index, test_index in kf.split(dataX):

    clf = RandomForestClassifier(n_estimators = 20 ,n_jobs = -1)

    clf = clf.fit(X[train_index], Y[train_index])

    yPred = clf.predict(X[test_index])

    # Calculating accuracy
    acc += accuracy_score(yPred, Y[test_index])


compare_value= acc/k
em[0,1] = acc/k
print ("Average Accuracy of RandomForest : " + str(acc/k) )
```

Average Accuracy of RandomForest : 0.76375

In [6]:
```python
from sklearn.naive_bayes import GaussianNB

kf = ShuffleSplit(n_splits=k, test_size=0.1, random_state=0)
acc = 0
for train_index, test_index in kf.split(X):

    clf = GaussianNB()

    clf = clf.fit(X[train_index], Y[train_index])
    yPred = clf.predict(X[test_index])

    # Calculating metrics
    acc += accuracy_score(yPred, Y[test_index])

print("Average Accuracy GaussianNB: " + str(acc/k) )
em[0,2] = acc/k
```

Average Accuracy GaussianNB: 0.705

Output_Files

In [7]:
```python
#Evaluation metric
columns = ["SVM","RandomForest","GaussianNB"]
rows = ["Accuracy"]
mycsv = pd.DataFrame(data=em,index=rows,columns=columns)
mycsv.to_csv("EvaluationMetric_10fold.csv")
print ("Created EvaluationMetric_10fold.csv file!")
```

Created EvaluationMetric_10fold.csv file!

In [8]:
```python
test = pd.read_csv('test.tsv', sep='\t')

clf = RandomForestClassifier(n_estimators = 20 ,n_jobs = -1)
clf = clf.fit(X, Y)
yPred = clf.predict(X)

testX = test.drop('Id',axis = 1)
testX = pd.get_dummies(testX)

predicted = clf.predict(testX)

output = np.zeros((len(predicted),2),dtype=object)

for i,j in zip(predicted,range(len(predicted))):
    if i == 1:
        output[j][0] = test.iloc[j]["Id"]
        output[j][1] = "Good"
    elif i == 2:
        output[j][0] = test.iloc[j]["Id"]
        output[j][1] = "Bad"

mycsv = pd.DataFrame(data=output,columns=["Client_ID","Predicted_Label"])
mycsv.to_csv("testSet_categories.csv", sep='\t',index=False, header=False)
print ("Created testSet_categories.csv file!")
```

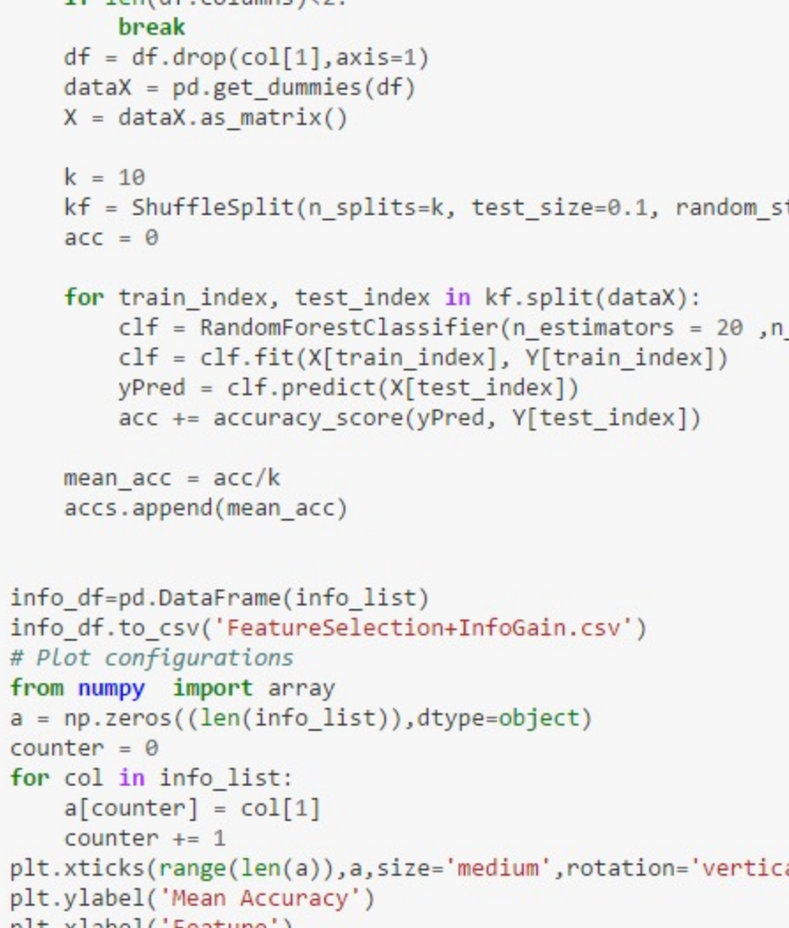
Created testSet_categories.csv file!

Information Gain & Accuracy

In [9]:
```python
import math
def my_entropy(data):
    goods=0
    Y=data['Label']
    for i in Y:
        if i==1:
            goods+=1
    p_good= goods/data.shape[0]
    p_bad= 1 - p_good
    s1=-(p_good * math.log(p_good))
    s2=-(p_bad * math.log(p_bad))
    s_fin = s1+s2
    return s_fin

def info_sum_attribute(arg):
    templist=list(set(df.columns))
    nonused_cols=[v for v in templist if not v.startswith('Label')]
    nonused_cols.remove(arg)
    newdf=df.drop(nonused_cols,axis=1)
    sum=0
    myvals=newdf[arg].value_counts()
    mycount=0
    for i in myvals.index:
        df_temp=newdf[newdf[arg]==i]
import math
def my_entropy(data):
    goods=0
    Y=data['Label']
    for i in Y:
        if i==1:
            goods+=1
    if data.shape[0]==0:
        return 0
    else:
        p_good= goods/data.shape[0]
    p_bad= 1 - p_good
    s1=-(p_good * math.log2(p_good))
    s2=-(p_bad * math.log2(p_bad))
    s_fin = s1+s2
    return s_fin

def info_sum_attribute(arg):
    templist=list(set(df.columns))
    nonused_cols=[v for v in templist if not v.startswith('Label')]
    nonused_cols.remove(arg)
    newdf=df.drop(nonused_cols,axis=1)
    sum=0
    myvals=newdf[arg].value_counts()
    mycount=0
    for i in myvals.index:
        df_temp=newdf[newdf[arg]==i]
        temp2=my_entropy(df_temp)
        amount_to_sum=(myvals[mycount]/newdf.shape[0]) * temp2
        sum+= amount_to_sum
        mycount+=1
    return sum

def info_num_sum(arg):
    templist=list(set(df.columns))
    nonused_cols=[v for v in templist if not v.startswith('Label')]
    nonused_cols.remove(arg)
    newdf=df.drop(nonused_cols,axis=1)
    sum=0
    (counters,bins)=np.histogram(newdf[arg],5)
    print(counters,bins)
    mycount=0
    j=0
    for i in counters:
        if j<4:
            df_temp=newdf[(newdf[arg]>=bins[j]) & (newdf[arg]<bins[j+1])]
        else:
            df_temp=newdf[(newdf[arg]>=bins[j]) & (newdf[arg]<=bins[j+1])]
        j+=1
        temp2=my_entropy(df_temp)
        amount_to_sum=(i/newdf.shape[0]) * temp2
        sum+= amount_to_sum
    return sum

info_list=[]

print("Entropy : "+ str(my_entropy(df)))
cols_to_drop=[v for v in cols_to_drop if not v.startswith('Label')]
catdata=df.drop(cols_to_drop,axis=1)
loopdata=catdata.drop('Label',axis=1)
for i in loopdata.columns:
  infogain=my_entropy(df)-info_sum_attribute(i)
  info_list.append([infogain,i])

cols_to_transform=[v for v in cols_to_transform if not v.startswith('Label')]
numericdata=df.drop(cols_to_transform,axis=1)
loopdata=numericdata.drop('Label',axis=1)
print("Counters & bins")
for i in loopdata.columns:
    infogain=my_entropy(df)-info_num_sum(i)
    info_list.append([infogain,i])
```

```
Entropy : 0.87975753726356
Counters & bins
[356 309  79  44  12] [  4.    17.6  31.2  44.8  58.4  72. ]
[547 165  56  22  10] [  250.   3389.   6528.   9667.  12806.  15945.]
[113 184   0 120 383] [ 1.   1.6  2.2  2.8  3.4  4. ]
[107 242   0 122 329] [ 1.   1.6  2.2  2.8  3.4  4. ]
[330 275 119  54  22] [ 19.   30.2  41.4  52.6  63.8  75. ]
[513 261   0  22   4] [ 1.   1.6  2.2  2.8  3.4  4. ]
[682   0   0   0 118] [ 1.   1.2  1.4  1.6  1.8  2. ]
```

In [10]:
```python
#info_list sorted based on info_gain
import operator
accs=[]
info_list=sorted(info_list,key=operator.itemgetter(0))

df = pd.read_csv('train.tsv', sep='\t')
tls=['Label','Id']
df=df.drop(tls,axis=1)


for col in info_list:
    if len(df.columns)<2:
        break
    df = df.drop(col[1],axis=1)
    dataX = pd.get_dummies(df)
    X = dataX.as_matrix()

    k = 10
    kf = ShuffleSplit(n_splits=k, test_size=0.1, random_state=0)
    acc = 0

    for train_index, test_index in kf.split(dataX):
        clf = RandomForestClassifier(n_estimators = 20 ,n_jobs = -1)
        clf = clf.fit(X[train_index], Y[train_index])
        yPred = clf.predict(X[test_index])
        acc += accuracy_score(yPred, Y[test_index])

    mean_acc = acc/k
    accs.append(mean_acc)


info_df=pd.DataFrame(info_list)
info_df.to_csv('FeatureSelection+InfoGain.csv')
# Plot configurations
from numpy import array
a = np.zeros((len(info_list)),dtype=object)
counter = 0
for col in info_list:
    a[counter] = col[1]
    counter += 1
plt.xticks(range(len(a)),a,size='medium',rotation='vertical')
plt.ylabel('Mean Accuracy')
plt.xlabel('Feature')
plt.plot(accs)

fig1 = plt.gcf()
plt.show()
plt.draw()

fig1.savefig('FeatureSelection')
```



In [11]:
```python
df = pd.read_csv('train.tsv', sep='\t')
Y = df['Label']
df = df.drop('Label',axis=1)
df = df.drop('Id',axis=1)

tmp=[]
for i in info_list:
    tmp.append(i[1])

if compare_value < max(accs):
    n = accs.index(max(accs))
    cols_to_drop = tmp[:n+1]
    df = df.drop(cols_to_drop,axis=1)
    test = pd.read_csv('test.tsv', sep='\t')
    test = test.drop(cols_to_drop,axis=1)

    dataX = pd.get_dummies(df)
    X = dataX.as_matrix()
    clf = RandomForestClassifier(n_estimators = 20 ,n_jobs = -1)
    clf = clf.fit(X, Y)
    yPred = clf.predict(X)

    testX = test.drop('Id',axis = 1)
    testX = pd.get_dummies(testX)

    predicted = clf.predict(testX)

    output = np.zeros((len(predicted),2),dtype=object)

    for i,j in zip(predicted,range(len(predicted))):
        if i == 1:
            output[j][0] = test.iloc[j]["Id"]
            output[j][1] = "Good"
        elif i == 2:
            output[j][0] = test.iloc[j]["Id"]
            output[j][1] = "Bad"

    mycsv = pd.DataFrame(data=output,columns=["Client_ID","Predicted_Label"])
    mycsv.to_csv("DropFeature_testSet_categories.csv", sep='\t',index=False, header=False)
    print ("Created testSet_categories.csv file!")
else:
    print("No DropFeature test set")
```

Created testSet_categories.csv file!

In [ ]: