In [8]:
```
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style("dark")
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
%pylab inline
%matplotlib inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Started by simply loading up the necessary libraries and data from the spreadsheets. Then I visualized the data we were given to see what the numbers looked like.

In [2]:
```
train = pd.read_csv('train.csv')
test  = pd.read_csv('test.csv')

target = train["label"]
train = train.drop("label",axis =1)
```

In [3]:
```
train = pd.read_csv('train.csv')
test  = pd.read_csv('test.csv')
X = train
y = train.pop("label")
```

In [9]:
```
figure(figsize(5,5))
for digit_num in range(0,64):
    subplot(8,8,digit_num+1)
    grid_data = train.iloc[digit_num].to_numpy().reshape(28,28)  # reshape from 1d to 2
    plt.imshow(grid_data, interpolation = "none", cmap = "bone_r")
    xticks([])
    yticks([])
```

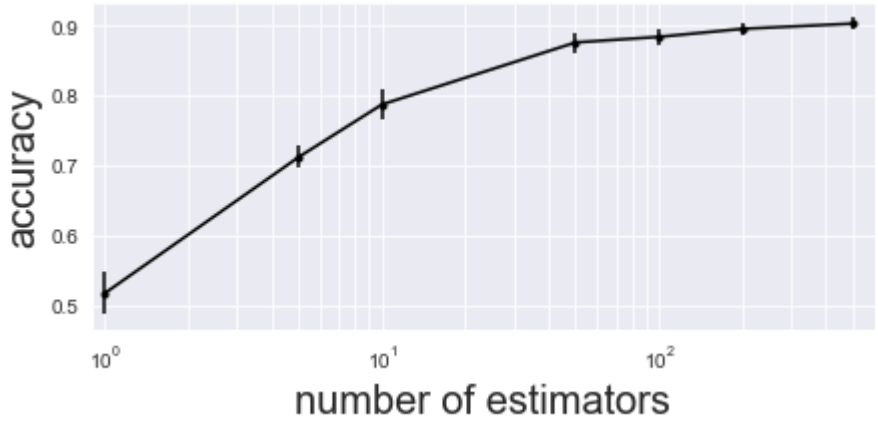
In [10]:
```
def evaluate_classifier(clf, data, target, split_ratio):
```

```
    trainX, testX, trainY, testY = train_test_split(data, target, train_size=split_rati
    clf.fit(trainX, trainY)
    return clf.score(testX,testY)
```

In [11]:
```
n_estimators_array = np.array([1,5,10,50,100,200,500])
n_samples = 10
n_grid = len(n_estimators_array)
score_array_mu =np.zeros(n_grid)
score_array_sigma = np.zeros(n_grid)
j=0
for n_estimators in n_estimators_array:
    score_array=np.zeros(n_samples)
    for i in range(0,n_samples):
        clf = RandomForestClassifier(n_estimators = n_estimators, n_jobs=1, criterion="
        score_array[i] = evaluate_classifier(clf, train.iloc[0:1000], target.iloc[0:100
    score_array_mu[j], score_array_sigma[j] = mean(score_array), std(score_array)
    j=j+1
```

Next I have a graph that looks at the amount of estimators in the Random Forest and how the accuracy increases depending on them. After that we can see that the PCA seperates the feature space into clusters for just two components. We then look at the components needed to see have a good variance percentage and its around 100ish components

In [12]:
```
figure(figsize(7,3))
errorbar(n_estimators_array, score_array_mu, yerr=score_array_sigma, fmt='k.-')
xscale("log")
xlabel("number of estimators",size = 20)
ylabel("accuracy",size = 20)
xlim(0.9,600)
grid(which="both")
```



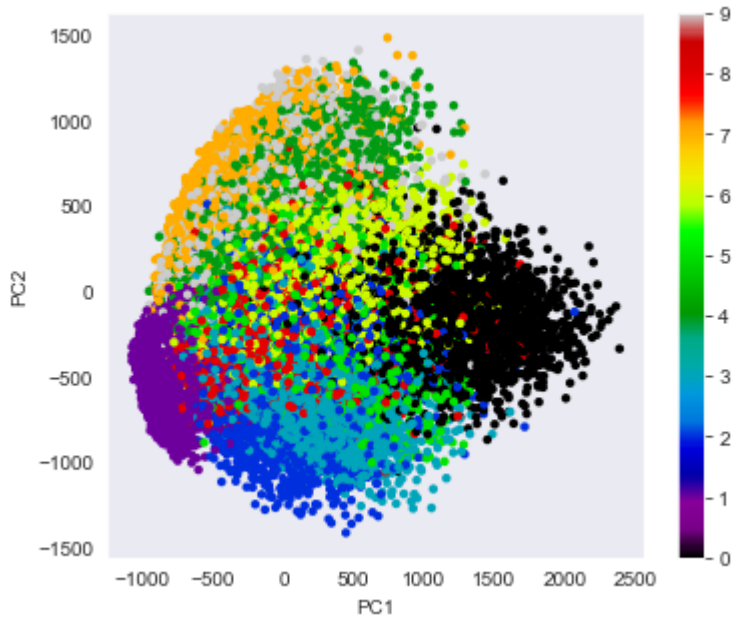
In [13]:
```
pca = PCA(n_components=2)
pca.fit(train)
transform = pca.transform(train)

figure(figsize(6,5))
plt.scatter(transform[:,0],transform[:,1], s=20, c = target, cmap = "nipy_spectral", ed
plt.colorbar()
clim(0,9)

xlabel("PC1")
ylabel("PC2")
```

```
Text(0, 0.5, 'PC2')
```

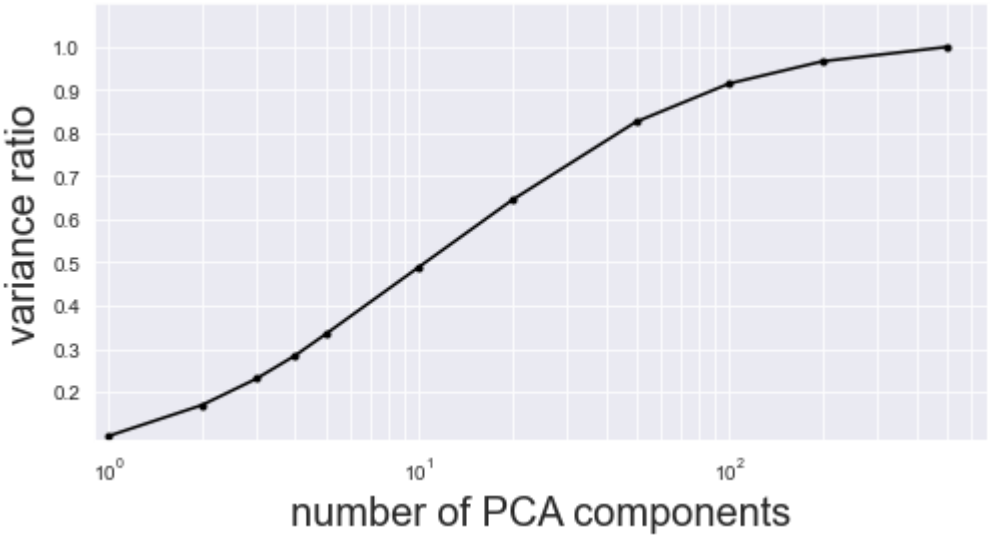
Out[13]:



In [14]:
```
n_components_array=([1,2,3,4,5,10,20,50,100,200,500])
vr = np.zeros(len(n_components_array))
i=0;
for n_components in n_components_array:
    pca = PCA(n_components=n_components)
    pca.fit(train)
    vr[i] = sum(pca.explained_variance_ratio_)
    i=i+1
```

In [15]:
```
figure(figsize(8,4))
plot(n_components_array,vr,'k.-')
xscale("log")
ylim(9e-2,1.1)
yticks(linspace(0.2,1.0,9))
xlim(0.9)
grid(which="both")
xlabel("number of PCA components",size=20)
ylabel("variance ratio",size=20)
```

Out[15]:
```
Text(0, 0.5, 'variance ratio')
```



Next I'm just using the random forest to see how accurate it is and it is about 96% accurate at guessing the correct number and that we can see that around 250 principle components are needed to explain the variance in our model.

In [16]:
```
from sklearn.model_selection import train_test_split
from time import time

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, shuffle=True,
```

In [17]:
```
from sklearn.ensemble import RandomForestClassifier
# Create model
rfc = RandomForestClassifier(random_state=9)
# Train
print("Training Random Forest Classifier")
t = time()
rfc.fit(X_train, y_train)
print(f"Finished training after {time()-t}s")
# Validation
t = time()
score = rfc.score(X_test, y_test)
print(f"Testing score: {score} in {time()-t}s")
```

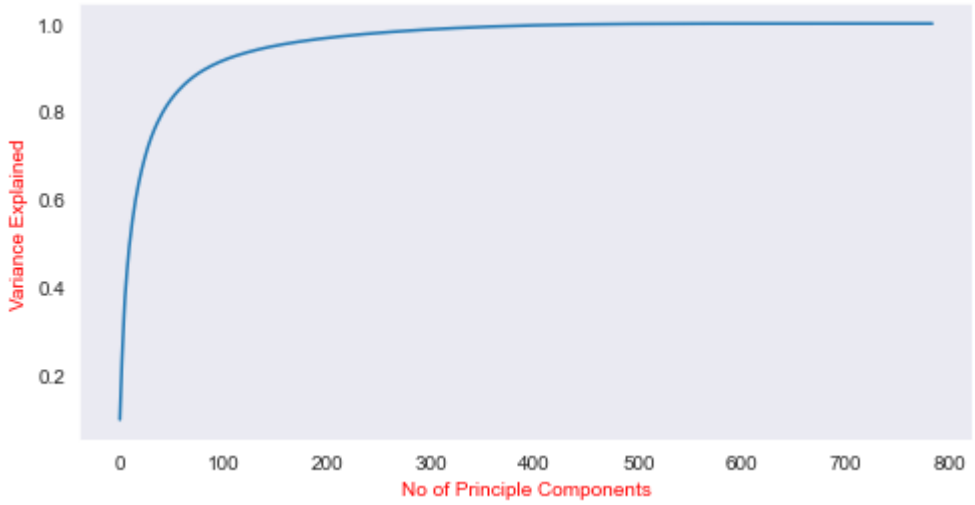
```
Training Random Forest Classifier
Finished training after 13.648204803466797s
Testing score: 0.9607503607503608 in 0.40415310859680176s
```

In [18]:
```
pca = PCA(n_components = 3)
X_train_trf = pca.fit_transform(X_train)
X_test_trf = pca.fit(X_test)

pca = PCA(n_components = None)
X_train_trf = pca.fit_transform(X_train)
X_test_trf = pca.fit(X_test)
```

In [19]:
```
plt. plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('No of Principle Components' ,color ='red')
plt.ylabel('Variance Explained' ,color ='red')
```

Out[19]:
```
Text(0, 0.5, 'Variance Explained')
```



In [20]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

standardized_scalar = StandardScaler()
standardized_data = standardized_scalar.fit_transform(train)
standardized_data.shape
```

Out[20]:
```
(42000, 784)
```

In [21]:
```
pca = PCA()
pca.n_components = 2
pca_data = pca.fit_transform(standardized_data)
pca_data.shape
pca_data
```

Out[21]:
```
array([[-5.14046186, -5.22601862],
       [19.29233558,  6.0337517 ],
       [-7.64450651, -1.70605525],
       ...,
       [ 0.49539109,  7.07698797],
       [ 2.30724285, -4.34503263],
       [-4.80765611,  1.55913765]])
```