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

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

In [3]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.cluster import KMeans
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
```

In [4]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

In [5]:
```
import xgboost as xgb
```

In [6]:
```
y = pd.read_csv('Dataset/actual.csv')
```

In [7]:
```
y = y.replace({'ALL':0,'AML':1})
labels = ['ALL', 'AML']
```

In [8]:
```
df_train = pd.read_csv('Dataset/data_set_ALL_AML_train.csv')


df_test = pd.read_csv('Dataset/data_set_ALL_AML_independent.csv')
```

In [9]:
```
train_to_keep = [col for col in df_train.columns if "call" not in col]
test_to_keep = [col for col in df_test.columns if "call" not in col]

X_train_tr = df_train[train_to_keep]
X_test_tr = df_test[test_to_keep]
```

In [10]:
```
train_columns_titles = ['Gene Description', 'Gene Accession Number', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10',
       '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25',
       '26', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '38']

X_train_tr = X_train_tr.reindex(columns=train_columns_titles)
```

In [11]:
```
test_columns_titles = ['Gene Description', 'Gene Accession Number','39', '40', '41', '42', '43', '44', '45', '46',
       '47', '48', '49', '50', '51', '52', '53',  '54', '55', '56', '57', '58', '59',
       '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72']

X_test_tr = X_test_tr.reindex(columns=test_columns_titles)
```

In [12]:
```
X_train = X_train_tr.T
X_test = X_test_tr.T
```

In [13]:
```
X_train.columns = X_train.iloc[1]
X_train = X_train.drop(["Gene Description", "Gene Accession Number"]).apply(pd.to_numeric)

X_test.columns = X_test.iloc[1]
X_test = X_test.drop(["Gene Description", "Gene Accession Number"]).apply(pd.to_numeric)
```

In [14]:
```
X_train = X_train.reset_index(drop=True)
y_train = y[y.patient <= 38].reset_index(drop=True)


X_test = X_test.reset_index(drop=True)
y_test = y[y.patient > 38].reset_index(drop=True)
```

In [15]:
```
X_train_fl = X_train.astype(float, 64)
X_test_fl = X_test.astype(float, 64)


scaler = StandardScaler()
X_train_scl = scaler.fit_transform(X_train_fl)
X_test_scl = scaler.transform(X_test_fl)
```

In [16]:
```
pca = PCA()
pca.fit_transform(X_train)
```

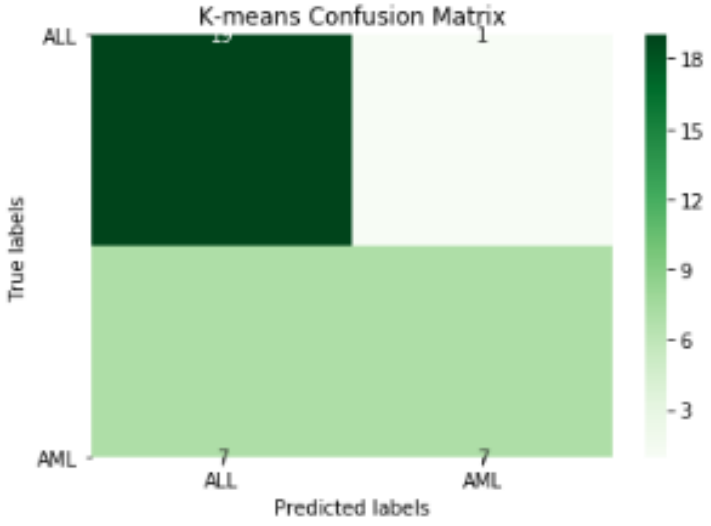
Out[16]:
```
array([[-4.12032149e+03,  8.43574289e+03, -1.39441668e+04, ...,
         2.51106855e+03,  3.92187680e+03,  1.22642865e-11],
       [ 1.86283598e+04,  1.44078238e+04,  1.66177453e+04, ...,
        -2.30960132e+02, -1.04099055e+03,  1.22642865e-11],
       [-1.58238732e+04,  1.40484268e+04,  4.73320627e+04, ...,
         5.48675197e+02, -2.26227734e+03,  1.22642865e-11],
       ...,
       [ 6.50848905e+04, -5.49595793e+04,  1.67854688e+04, ...,
         1.18708820e+01, -1.47894896e+03,  1.22642865e-11],
       [ 4.97670530e+04, -3.81956823e+04,  2.93511865e+03, ...,
         2.66462156e+03,  7.99461277e+02,  1.22642865e-11],
       [ 1.08241948e+04, -1.68550421e+04, -9.46017931e+02, ...,
        -2.04773331e+03, -1.96917341e+03,  1.22642865e-11]])
```

In [17]:
```
total = sum(pca.explained_variance_)
k = 0
current_variance = 0
while current_variance/total < 0.90:
    current_variance += pca.explained_variance_[k]
    k = k + 1


pca = PCA(n_components=k)
X_train.pca = pca.fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

In [18]:

```
#1st implementation algorithm- k means
kmeans = KMeans(n_clusters=2, random_state=0).fit(X_train_scl)
km_pred = kmeans.predict(X_test_scl)

print('K-means accuracy:', round(accuracy_score(y_test.iloc[:,1], km_pred), 3))

cm_km = confusion_matrix(y_test.iloc[:,1], km_pred)

ax = plt.subplot()
sns.heatmap(cm_km, annot=True, ax = ax, fmt='g', cmap='Greens')


ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('K-means Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels, rotation=360);
```

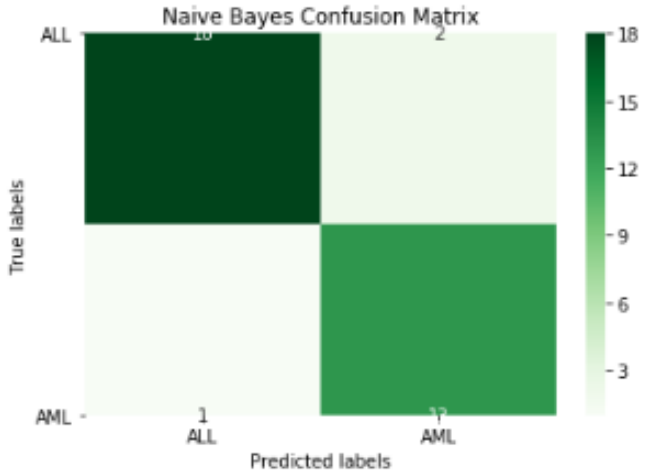
```
K-means accuracy: 0.765
```



In [19]:

```
#2nd implementation algorithm-naive bayes
nb_model = GaussianNB()

nb_model.fit(X_train, y_train.iloc[:,1])

nb_pred = nb_model.predict(X_test)

print('Naive Bayes accuracy:', round(accuracy_score(y_test.iloc[:,1], nb_pred), 3))

cm_nb =  confusion_matrix(y_test.iloc[:,1], nb_pred)

ax = plt.subplot()
sns.heatmap(cm_nb, annot=True, ax = ax, fmt='g', cmap='Greens')


ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Naive Bayes Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels, rotation=360);
```

```
Naive Bayes accuracy: 0.912
```

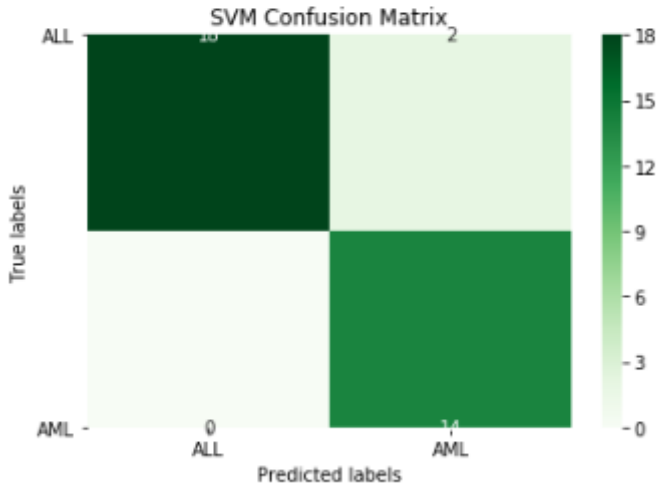


In [21]:
```
#3rd implementation algorithm-support vector machines

svm_param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001, 0.00001, 10], "kernel": ["linear", "rbf", "p

svm_grid = GridSearchCV(SVC(), svm_param_grid, cv=3)

svm_grid.fit(X_train_pca, y_train.iloc[:,1])


best_svc = svm_grid.best_estimator_


svm_pred = best_svc.predict(X_test_pca)
print('SVM accuracy:', round(accuracy_score(y_test.iloc[:,1], svm_pred), 3))

cm_svm =  confusion_matrix(y_test.iloc[:,1], svm_pred)

ax = plt.subplot()
sns.heatmap(cm_svm, annot=True, ax = ax, fmt='g', cmap='Greens')


ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('SVM Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels, rotation=360);
```

```
SVM accuracy: 0.941
```

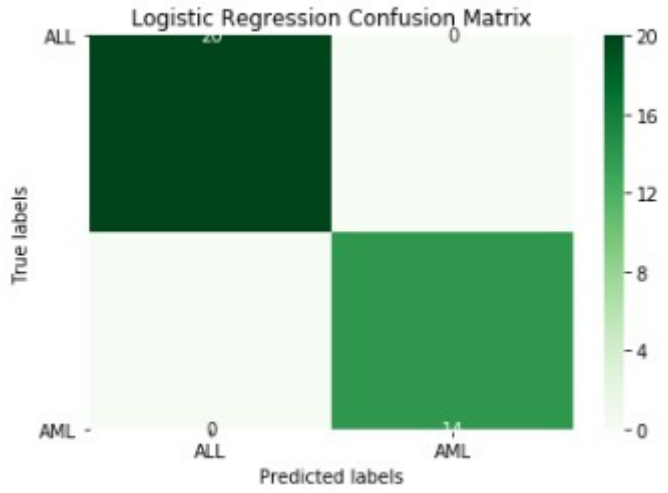
SVM accuracy: 0.941

```
C:\Anaconda\lib\site-packages\sklearn\model_selectio
ll change from True to False in version 0.22 and wil
are unequal.
  DeprecationWarning)
```



In [22]:
```
#4th implementation alogrithm-Logistic regression
log_grid = {'C': [1e-03, 1e-2, 1e-1, 1, 10],
                  'penalty': ['l1', 'l2']}

log_estimator = LogisticRegression(solver='liblinear')

log_model = GridSearchCV(estimator=log_estimator,
                   param_grid=log_grid,
                   cv=3,
                   scoring='accuracy')

log_model.fit(X_train, y_train.iloc[:,1])



best_log = log_model.best_estimator_


log_pred = best_log.predict(X_test)

print('Logistic Regression accuracy:', round(accuracy_score(y_test.iloc[:,1], log_pred), 3))

cm_log =  confusion_matrix(y_test.iloc[:,1], log_pred)

ax = plt.subplot()
sns.heatmap(cm_log, annot=True, ax = ax, fmt='g', cmap='Greens')

ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Logistic Regression Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels, rotation=360);
```

```
Logistic Regression accuracy: 1.0
```



```
#5th implementation-XG Boost
xgb3_model = xgb.XGBClassifier()
xgb3_model.fit(X_train, y_train.iloc[:,1])

xgb3_pred = xgb3_model.predict(X_test)

print('XGB accuracy:', round(accuracy_score(y_test.iloc[:,1], xgb3_pred), 3))

cm_xgb3 = confusion_matrix(y_test.iloc[:,1], xgb3_pred)

ax = plt.subplot()
sns.heatmap(cm_xgb3, annot=True, ax = ax, fmt='g', cmap='Greens')


ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('XGB Confusion Matrix')
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels, rotation=360);
```

```
XGB accuracy: 0.912
```

XGB accuracy: 0.912