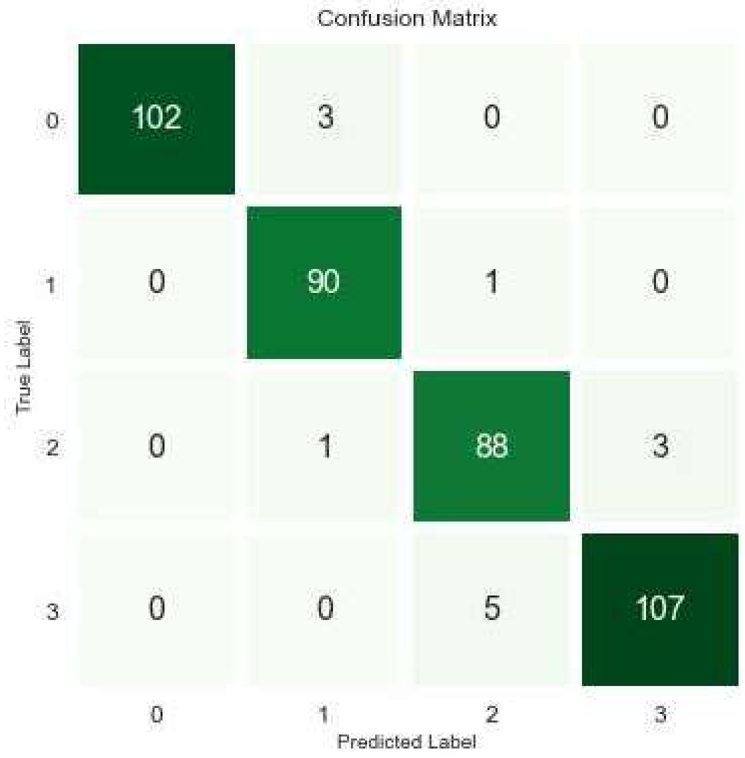
In [33]:

```
cm = confusion_matrix(y_test, y_pred_lda)

df1 = pd.DataFrame(columns=["0","1","2","3"], index= ["0","1","2","3"], data= cm )

f,ax = plt.subplots(figsize=(6,6))

sns.heatmap(df1, annot=True,cmap="Greens", fmt= '.0f',ax=ax,linewidths = 5, cbar = Fals
plt.xlabel("Predicted Label")
plt.xticks(size = 12)
plt.yticks(size = 12, rotation = 0)
plt.ylabel("True Label")
plt.title("Confusion Matrix", size = 12)
plt.show()
```



Hyperparameter Tuning - Grid Search - Cross Validation

```

# We will compare 8 classifier and evaluate mean accuracy of each of them by stratified cross validation.

Decision Tree Classifier
SVC
Random Forest Classifier
Logistic Regression
KNN Classifier
Stochastic Gradient Descent Classifier
Gradient Boosting Classifier
LightGBM Classifier
```

In [34]:

```
classifier = [DecisionTreeClassifier(random_state = random_state),
              SVC(random_state = random_state, probability = True),
              RandomForestClassifier(random_state = random_state),
              LogisticRegression(random_state = random_state),
              KNeighborsClassifier(),
              SGDClassifier(random_state = random_state),
              GradientBoostingClassifier(random_state = random_state),
              LGBMClassifier(random_state = random_state)]

dt_param_grid = {"min_samples_split" : range(10,500,20),
                 "max_depth": range(1,20,2)}

svc_param_grid = {"kernel" : ["rbf"],
                  "gamma": [0.001, 0.01, 0.1, 1],
                  "C": [1,10,50,100,200,300,1000]}

rf_param_grid = {"max_features": [1,3,10],
                 "min_samples_split":[2,3,10],
                 "min_samples_leaf":[1,3,10],
                 "bootstrap":[False],
                 "n_estimators":[100,300],
                 "criterion":["gini"]}

logreg_param_grid = {"C":np.logspace(-4, 4, 20),
                     "penalty": ["l1","l2","none"]}

knn_param_grid = {"n_neighbors": np.linspace(2,20,12, dtype = int).tolist(),
                  "weights": ["uniform","distance"],
                  "metric":["euclidean","manhattan","minkowski"],
                  "leaf_size": [30]}

sgdc_param_grid = {
    "loss" : ["hinge", "log", "squared_hinge", "modified_huber"],
    "alpha" : [0.0001, 0.001, 0.01, 0.1],
    "penalty" : ["l2", "l1", "none"]}

gbc_param_grid = {
    "learning_rate": [0.05, 0.1, 0.2],
    "min_samples_split": [2,3,10],
    "min_samples_leaf": [1,3,10]
    }


lgbmc_param_grid = {
    'num_leaves': [31, 127],
    'reg_alpha': [0.1, 0.5]}


classifier_param = [dt_param_grid,
                    svc_param_grid,
                    rf_param_grid,
                    logreg_param_grid,
                    knn_param_grid,
                    sgdc_param_grid,
                    gbc_param_grid,
                    lgbmc_param_grid]

cv_result = []
best_estimators = []
```

```
mean_squared_errors = []
roc_auc_scores = []
recall_scores = []
precision_scores = []
f1_scores = []


for i in range(len(classifier)):
    print("---------------------------------------------------------------------------
    clf = GridSearchCV(classifier[i],
                       param_grid=classifier_param[i],
                       cv = StratifiedKFold(n_splits = 10),
                       scoring = "accuracy",
                       n_jobs = -1,verbose = 2)

    clf.fit(X_train,y_train)

    cv_result.append(clf.best_score_)

    mean_squared_errors.append(mean_squared_error(y_test,clf.predict(X_test)))

    roc_auc_scores.append(roc_auc_score(y_test, clf.predict_proba(X_test), multi_class

    recall_scores.append(recall_score(y_test, clf.predict(X_test), average='weighted')

    precision_scores.append(precision_score(y_test, clf.predict(X_test), average='weig

    f1_scores.append(f1_score(y_test, clf.predict(X_test), average='weighted'))

    best_estimators.append(clf.best_estimator_)

    print("Model: {}".format(classifier[i]))
    print("Accuracy: %{}".format(round(cv_result[i]*100,2)))
    print("MSE: {}".format(mean_squared_errors[i]))
    print("ROC AUC: {}".format(roc_auc_scores[i]))
    print("Recall: {}".format(recall_scores[i]))
    print("Precision: {}".format(precision_scores[i]))
    print("F1-Score: {}".format(f1_scores[i]))
    print("Best Estimator: {}".format(clf.best_estimator_))

print("---------------------------------------------------------------------------")

sns.set_style("darkgrid")
cv_results = pd.DataFrame({"Accuracy":cv_result,
                           "MSE":mean_squared_errors,
                           "ROC AUC":roc_auc_scores,
                           "Recall": recall_scores,
                           "Precision": precision_scores,
                           "F1-Score":f1_scores,
                           "Models":["DecisionTreeClassifier",
                                     "SVC",
                                     "RandomForestClassifier",
                                     "LogisticRegression",
                                     "KNeighborsClassifier",
                                     "SGDClassifier",
                                     "GBClassifier",
                                     "LGBMClassifier"]})

cv_results.index = cv_results["Models"]

cv_results  = cv_results.drop(["Models"], axis = 1)
```

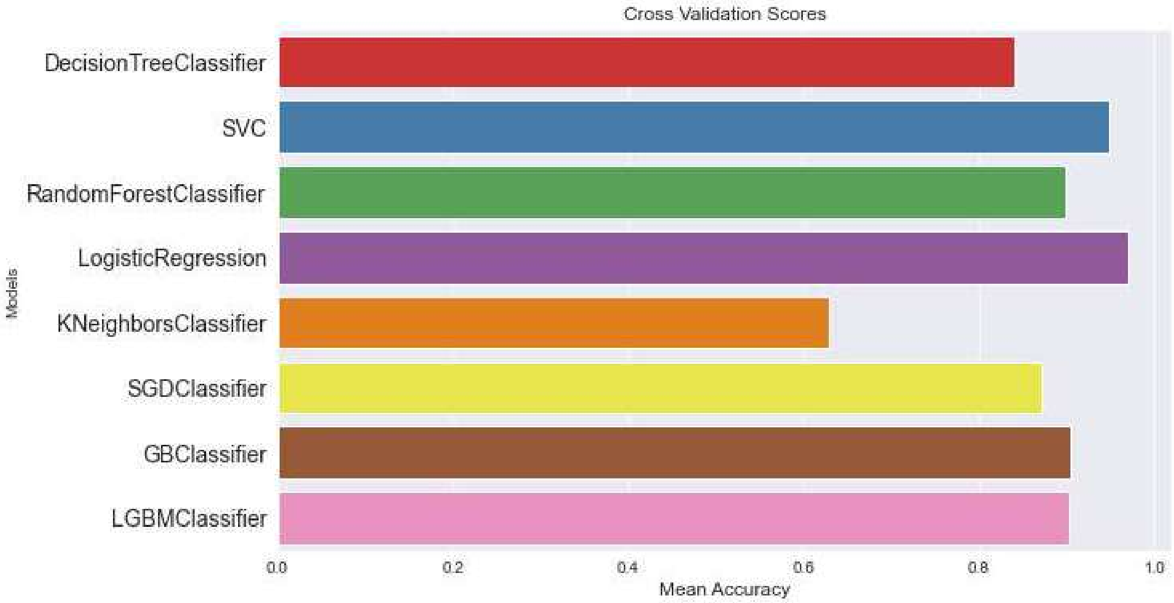
```

f,ax = plt.subplots(figsize=(14,10))

sns.heatmap(cv_results, annot=True,cmap = "Blues",fmt= '.3f',
            ax=ax,linewidths = 5, cbar = False,
            annot_kws={"size": 18})

plt.xticks(size = 18)
plt.yticks(size = 18, rotation = 0)
plt.ylabel("Models")
plt.title("Grid Search Results", size = 16)
plt.show()
```

```
--------------------------------------------------------------------------
-
Fitting 10 folds for each of 250 candidates, totalling 2500 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent worker
s.
[Parallel(n_jobs=-1)]: Done  72 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 2408 tasks      | elapsed:    5.4s
[Parallel(n_jobs=-1)]: Done 2500 out of 2500 | elapsed:    5.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent worker
s.

Model: DecisionTreeClassifier(random_state=42)
Accuracy: %84.06
MSE: 0.175
ROC AUC: 0.911858935204842
Recall: 0.825
Precision: 0.8276565656565658
F1-Score: 0.8257884190113292
```

Cross Validation Scores

In [35]:

```
sns.set_style("darkgrid")
cv_results = pd.DataFrame({"Cross Validation Means":cv_result,
                           "Models":["DecisionTreeClassifier", "SVC",
                                     "RandomForestClassifier",
                                     "LogisticRegression",
                                     "KNeighborsClassifier",
                                     "SGDClassifier",
                                     "GBClassifier",
                                     "LGBMClassifier"]})

plt.figure(figsize = (10,6))
sns.barplot("Cross Validation Means", "Models",
            data = cv_results, palette = "Set1")
plt.xlabel("Mean Accuracy",
           size = 12)
plt.yticks(size = 14)
plt.title("Cross Validation Scores",
          size = 12)
plt.show()
```



Ensemble Learning