Using Gradient Boosting Decision Trees on DonorsChoose Dataset.

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpus/stopwords.zip.
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package maxnet_ipa to /root/nltk_data...
[nltk_data]   Unzipping models/maxnet_ipa.zip.
[nltk_data] Downloading package cmudict to /root/nltk_data...
[nltk_data]   Unzipping corpus/cmudict.zip.
```

```
# Loading data.
project_data = pd.read_csv('train.csv')
# Replacing the column names with the names given in the description.
project_data.columns = ['id', 'teacher_id', 'teacher_prefix', 'school_state',
                        'project_submitted_datetime', 'project_grade_category',
                        'project_title', 'project_essay_1', 'project_essay_2',
                        'project_essay_3', 'project_essay_4', 'project_resource_summary',
                        'teacher_number_of_previously_posted_projects', 'project_is_approved']
# Checking the shape of the data.
print("Number of data points in train data", project_data.shape)
print("-"*50)
# Checking the column names.
print("The column names", project_data.columns)
```

```
Number of data points in train data (109248, 16)
--------------------------------------------------
The column names Index(['id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category',
       'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved'],
      [redacted]
```

```
# Checking the first 5 rows of the data.
project_data.head(5)
```

id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1	project_essay_2	project_essay_3	project_essay_4	project_resource_summary	teacher_number_of_previously_posted_projects	project_is_approved
[redacted]													

Using TFIDF to encode the feature 'essay'

```
# Using this function to convert the text data into numerical data.
# Fit the vocabulary and the inverse document frequencies.
# Transform documents to document-term matrix.
tfidf_essay = tfidf_vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encoding ",tfidf_essay.shape)
```

```
Shape of matrix after one hot encoding  (109248, 5000)
```

Calculating the Sentiment scores from nltk library for feature 'essay'

```
# Import the library.
from nltk.sentiment.vader import SentimentIntensityAnalyzer
[redacted]
# Calculating the sentiment scores from nltk library for feature 'essay'.
sid = SentimentIntensityAnalyzer()
[redacted]
# Calculating the positive, negative, neutral and compound sentiment from the feature essay.
# https://stackoverflow.com/questions/40325546/how-to-add-new-features-to-sklearn-pipeline
[redacted]
```

```
[redacted]
[nltk_data] Error loading punkt: [redacted]
[nltk_data] Error loading wordnet: [redacted]
[nltk_data] Error loading averaged_perceptron_tagger: [redacted]
[nltk_data] Error loading maxnet_ipa: [redacted]
[nltk_data] Error loading cmudict: [redacted]
[redacted]
[nltk_data] Error loading punkt_ipa: [redacted]
[redacted]
[nltk_data] Error loading vader_lexicon: [redacted]
[redacted]
[nltk_data] Error loading stopwords
```

In [49]:
```
# Analyze the train and test scores with hyperparameters
train_sizes, train_scores, test_scores = learning_curve(DecisionTreeClassifier(max_depth=3, random_state=42),
                                                        X_train, y_train, cv=5, n_jobs=-1,
                                                        train_sizes=np.linspace(0.1, 1.0, 10), scoring='accuracy')

# Calculate mean and standard deviation for training and test scores
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, label='Training score')
plt.plot(train_sizes, test_mean, label='Cross-validation score')
plt.show()
```


In [50]:
```
print(best_parameters)
```

Out[50]:
```
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

In [51]:
```
# Create a DecisionTreeClassifier with the best parameters
model = DecisionTreeClassifier(criterion='gini', max_depth=3, min_samples_leaf=1, min_samples_split=2, random_state=42)

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict the target variable for the test data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy
print("Accuracy:", accuracy)
```

Out[51]:
```
Accuracy: 0.9
```

In [52]:
```
# Create a learning curve
train_sizes, train_scores, test_scores = learning_curve(model, X, y, cv=5)

# Calculate mean and standard deviation for training and test scores
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot the learning curve
plt.plot(train_sizes, train_mean, label='Training score')
plt.plot(train_sizes, test_mean, label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Accuracy')
plt.title('Learning Curve')
plt.legend()
plt.show()
```


Confusion Matrix

In [53]:
```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

In [54]:
```
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

Out[54]:
```
Confusion Matrix:
[[ 50  0  0]
 [  0 50  0]
 [  0  0 50]]
```


In [55]:
```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

Out[55]:
```
Confusion Matrix:
[[ 50  0  0]
 [  0 50  0]
 [  0  0 50]]
```

In [56]:
```
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```


Summary

Algorithm	Hyperparameters	Accuracy	Train Acc	Test Acc
GBDT with PCAAF	n_estimators=50, learning_rate=0.1, max_depth=3, n_jobs=-1, random_state=42	0.9667	0.700	0.767
GBDT with PCAAF RFE	n_estimators=50, learning_rate=0.1, max_depth=3, n_jobs=-1, random_state=42	0.9667	0.700	0.700