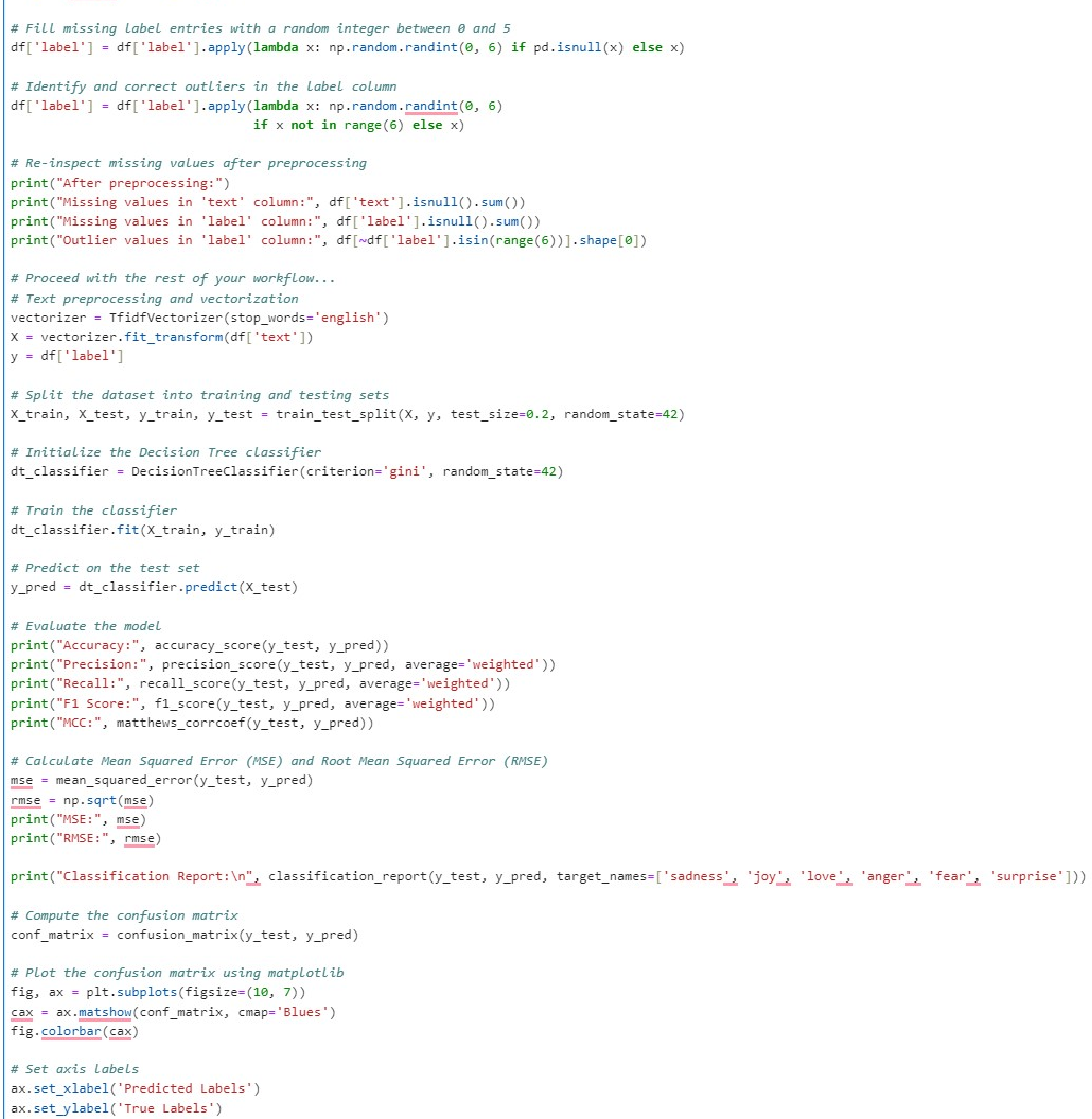

```python
# Fill missing label entries with a random integer between 0 and 5
df['label'] = df['label'].apply(lambda x: np.random.randint(0, 6) if pd.isnull(x) else x)

# Identify and correct outliers in the label column
df['label'] = df['label'].apply(lambda x: np.random.randint(0, 6)
                                if x not in range(6) else x)

# Re-inspect missing values after preprocessing
print("After preprocessing:")
print("Missing values in 'text' column:", df['text'].isnull().sum())
print("Missing values in 'label' column:", df['label'].isnull().sum())
print("Outlier values in 'label' column:", df[~df['label'].isin(range(6))].shape[0])

# Proceed with the rest of your workflow...
# Text preprocessing and vectorization
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['text'])
y = df['label']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Decision Tree classifier
dt_classifier = DecisionTreeClassifier(criterion='gini', random_state=42)

# Train the classifier
dt_classifier.fit(X_train, y_train)

# Predict on the test set
y_pred = dt_classifier.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print("MCC:", matthews_corrcoef(y_test, y_pred))

# Calculate Mean Squared Error (MSE) and Root Mean Squared Error (RMSE)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("MSE:", mse)
print("RMSE:", rmse)

print("Classification Report:\n", classification_report(y_test, y_pred, target_names=['sadness', 'joy', 'love', 'anger', 'fear', 'surprise']))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix using matplotlib
fig, ax = plt.subplots(figsize=(10, 7))
cax = ax.matshow(conf_matrix, cmap='Blues')
fig.colorbar(cax)

# Set axis labels
ax.set_xlabel('Predicted Labels')
ax.set_ylabel('True Labels')
```