```
# Load CIFAR-10 dataset and preprocess

import numpy as np
from tensorflow.keras.datasets import cifar10

# Load the dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Flatten label arrays
y_train = y_train.flatten()
y_test = y_test.flatten()

# Normalize pixel values (0-255 to 0-1)
X_train = X_train / 255.0
X_test = X_test / 255.0

# Flatten images for Random Forest (from 32x32x3 to 3072 features)
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Print shapes to confirm
print("Training data shape:", X_train_flat.shape)
print("Training labels shape:", y_train.shape)
print("Test data shape:", X_test_flat.shape)
print("Test labels shape:", y_test.shape)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Training data shape: (50000, 3072)
Training labels shape: (50000,)
Test data shape: (10000, 3072)
Test labels shape: (10000,)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Use a smaller training sample for faster tuning
X_train_small = X_train_flat[:5000]
y_train_small = y_train[:5000]

# Smaller parameter grid
param_grid = {
    'n_estimators': [50],
    'max_depth': [10, None],
    'min_samples_split': [2],
    'min_samples_leaf': [1]
}

# Initialize model
```

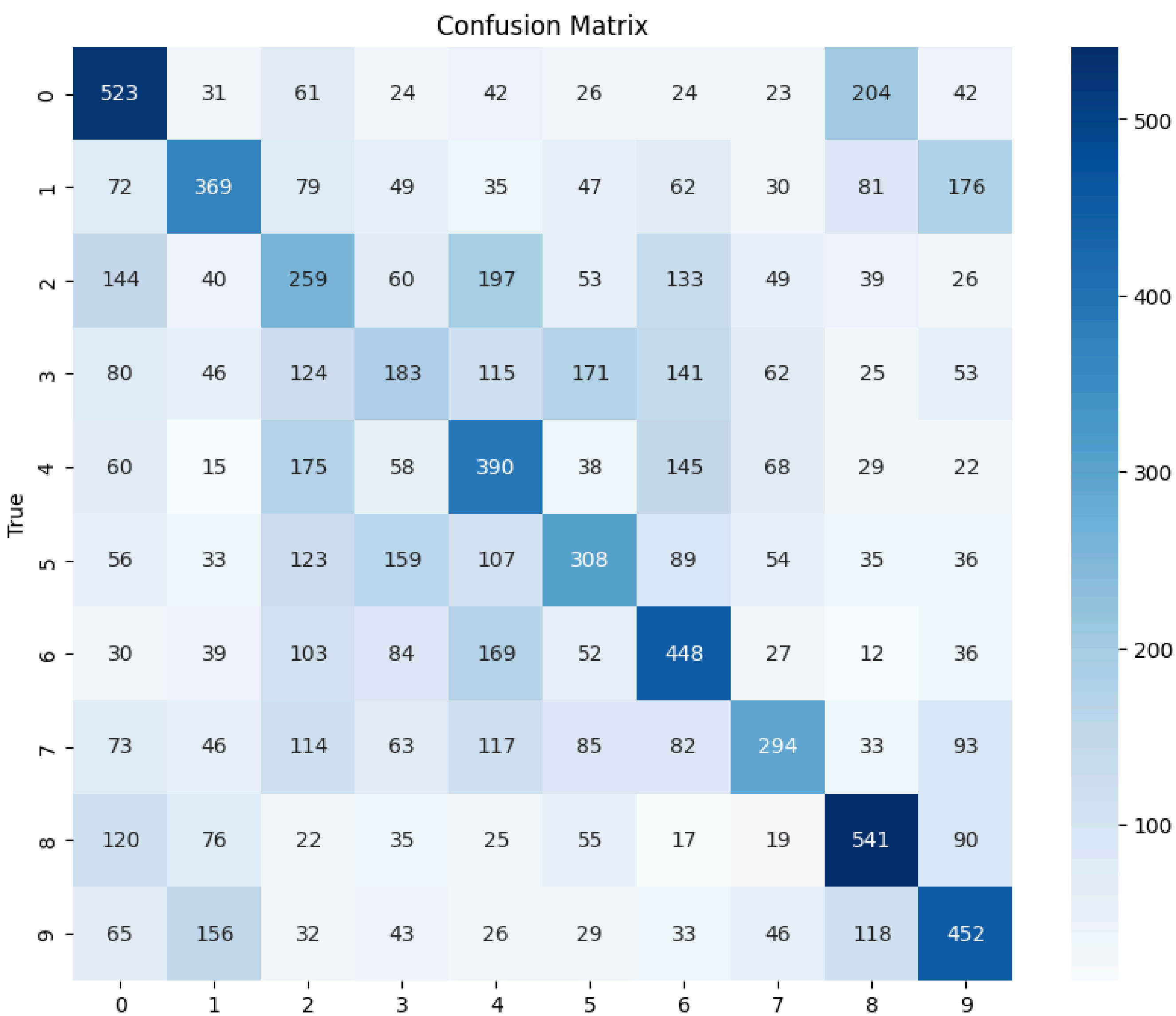
```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)

# Grid Search with 3-fold cross-validation
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, verbose=2,

# Fit the model
grid_search.fit(X_train_small, y_train_small)

# Best parameters
print("Best parameters found:", grid_search.best_params_)

# Best estimator
best_rf = grid_search.best_estimator_
```

```
Fitting 3 folds for each of 2 candidates, totalling 6 fits
Best parameters found: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2
```

```
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, classification_
import seaborn as sns
import matplotlib.pyplot as plt

# Predict on the test data
y_pred = best_rf.predict(X_test_flat)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Precision, Recall, F1-Score
precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, average='weightec

# Print scores
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")

# Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
```

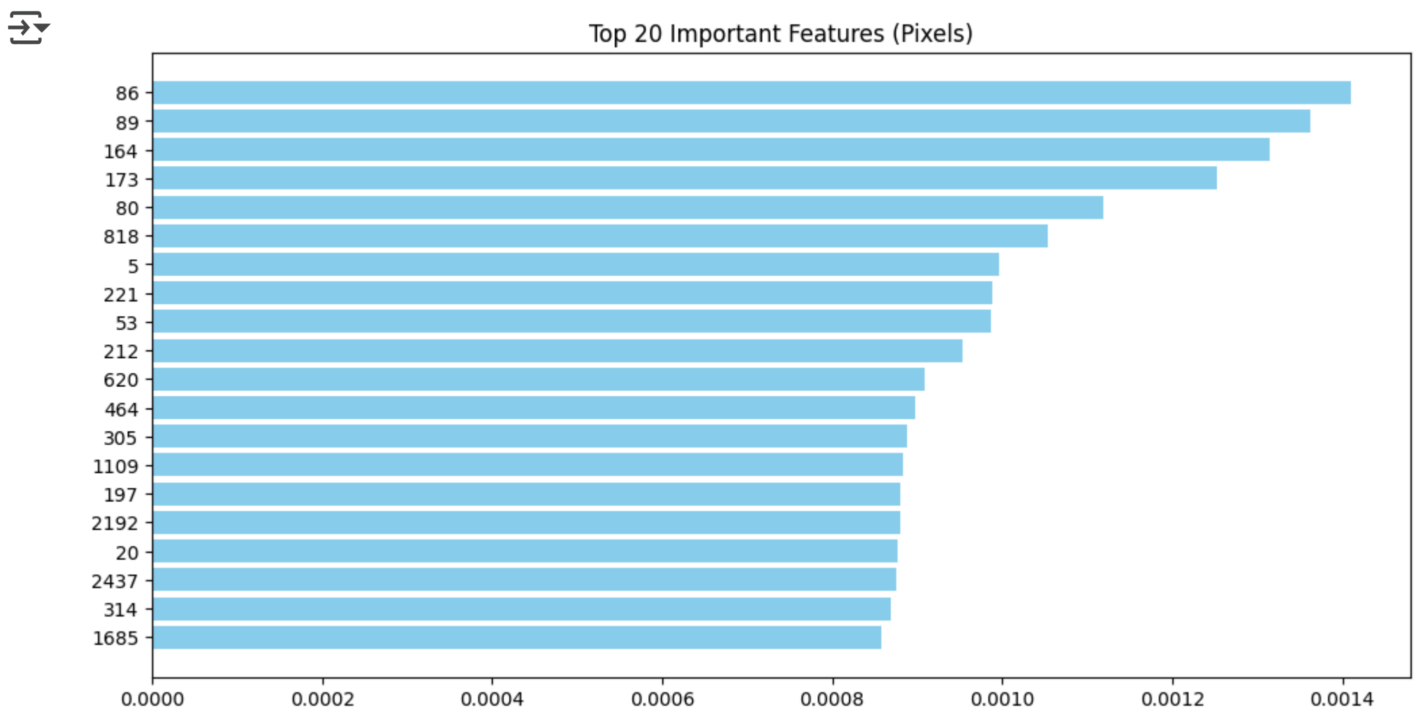
```
plt.show()
```

```
Accuracy: 0.3767
Precision: 0.3759
Recall: 0.3767
F1 Score: 0.3728

Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.52      0.47      1000
           1       0.43      0.37      0.40      1000
           2       0.24      0.26      0.25      1000
           3       0.24      0.18      0.21      1000
           4       0.32      0.39      0.35      1000
           5       0.36      0.31      0.33      1000
           6       0.38      0.45      0.41      1000
           7       0.44      0.29      0.35      1000
           8       0.48      0.54      0.51      1000
           9       0.44      0.45      0.45      1000

    accuracy                           0.38     10000
   macro avg       0.38      0.38      0.37     10000
weighted avg       0.38      0.38      0.37     10000
```



Predicted

```
# Feature Importance Visualization

importances = best_rf.feature_importances_

# Get top 20 important features (pixels)
top_indices = importances.argsort()[-20:]

# Plot
plt.figure(figsize=(12,6))
plt.barh(range(20), importances[top_indices], color='skyblue')
plt.yticks(range(20), top_indices)
plt.xlabel('Feature Importance')
plt.title('Top 20 Important Features (Pixels)')
plt.show()
```




```
import cv2

# Preprocessing function for new image
def preprocess_image(img_path):
    """
    Load and preprocess image for prediction
    """
    img = cv2.imread(img_path)
    if img is None:
        raise ValueError("Image not found or can't be opened.")

    img_resized = cv2.resize(img, (32, 32))
    img_norm = img_resized / 255.0
```

```
    img_flat = img_norm.flatten().reshape(1, -1)
    return img_flat

# Prediction function
def predict_new_image(img_path, model=best_rf):
    img_processed = preprocess_image(img_path)
    prediction = model.predict(img_processed)
    return prediction[0]
```

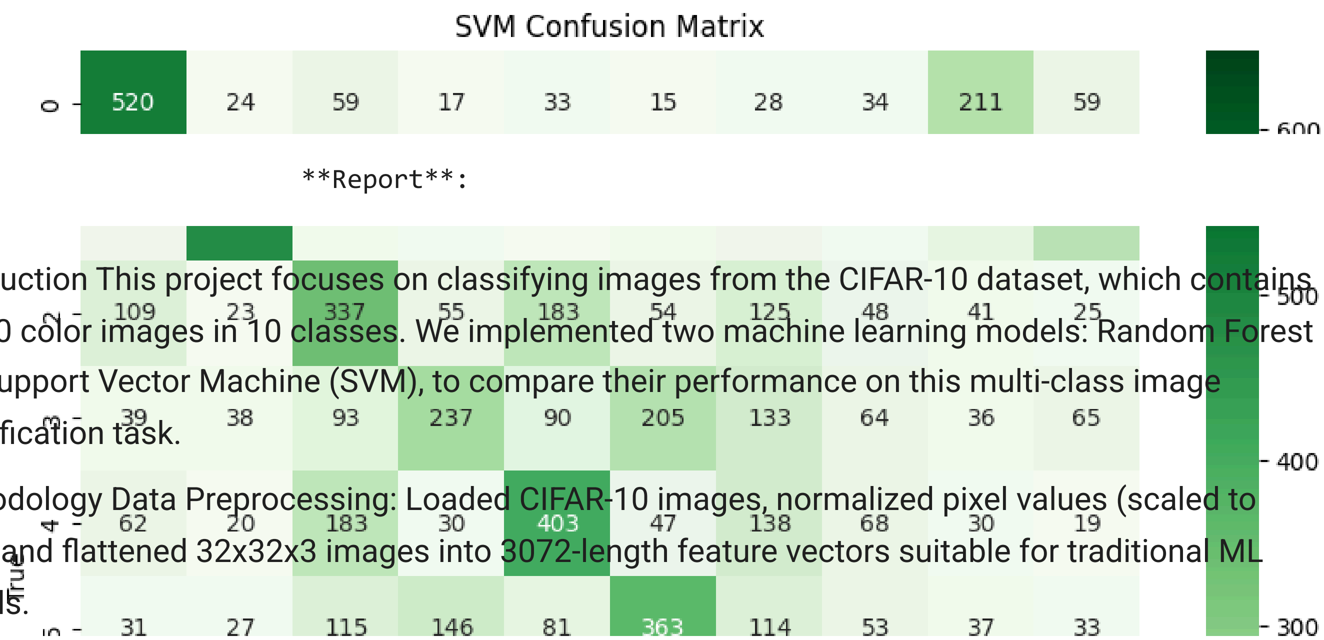
```
img_path = 'converted_image.jpg'  # your uploaded image filename
predicted_class = predict_new_image(img_path)

# CIFAR-10 class labels
cifar10_labels = {
    0: 'airplane',
    1: 'automobile',
    2: 'bird',
    3: 'cat',
    4: 'deer',
    5: 'dog',
    6: 'frog',
    7: 'horse',
    8: 'ship',
    9: 'truck'
}

print(f"Predicted class: {predicted_class}")
print(f"Predicted label: {cifar10_labels[predicted_class]}")
```

```
Predicted class: 3
Predicted label: cat
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import time

# Use smaller sample for faster training like before
X_train_small = X_train_flat[:5000]
y_train_small = y_train[:5000]

# Initialize SVM with default parameters (you can tune later)
svm = SVC(kernel='rbf', random_state=42)

# Measure training time
start_time = time.time()
svm.fit(X_train_small, y_train_small)
end_time = time.time()
print(f"SVM training time: {end_time - start_time:.2f} seconds")
```

```
# Predict on test set
y_pred_svm = svm.predict(X_test_flat)

# Evaluate
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f"SVM Accuracy: {accuracy_svm:.4f}")

print("\nClassification Report (SVM):")
print(classification_report(y_test, y_pred_svm))

# Optional: Confusion matrix plot
import seaborn as sns
import matplotlib.pyplot as plt

cm_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(10,8))
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Greens')
plt.title("SVM Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

```
SVM training time: 62.01 seconds
SVM Accuracy: 0.4435

Classification Report (SVM):
              precision    recall  f1-score   support

           0       0.51      0.52      0.52      1000
           1       0.55      0.48      0.51      1000
           2       0.33      0.34      0.33      1000
           3       0.34      0.24      0.28      1000
           4       0.36      0.40      0.38      1000
           5       0.38      0.36      0.37      1000
           6       0.43      0.52      0.47      1000
           7       0.52      0.41      0.46      1000
           8       0.53      0.65      0.58      1000
           9       0.47      0.53      0.50      1000

    accuracy                           0.44     10000
   macro avg       0.44      0.44      0.44     10000
weighted avg       0.44      0.44      0.44     10000
```



Report:

Introduction This project focuses on classifying images from the CIFAR-10 dataset, which contains 60,000 color images in 10 classes. We implemented two machine learning models: Random Forest and Support Vector Machine (SVM), to compare their performance on this multi-class image classification task.

Methodology Data Preprocessing: Loaded CIFAR-10 images, normalized pixel values (scaled to [0,1]), and flattened 32x32x3 images into 3072-length feature vectors suitable for traditional ML models.