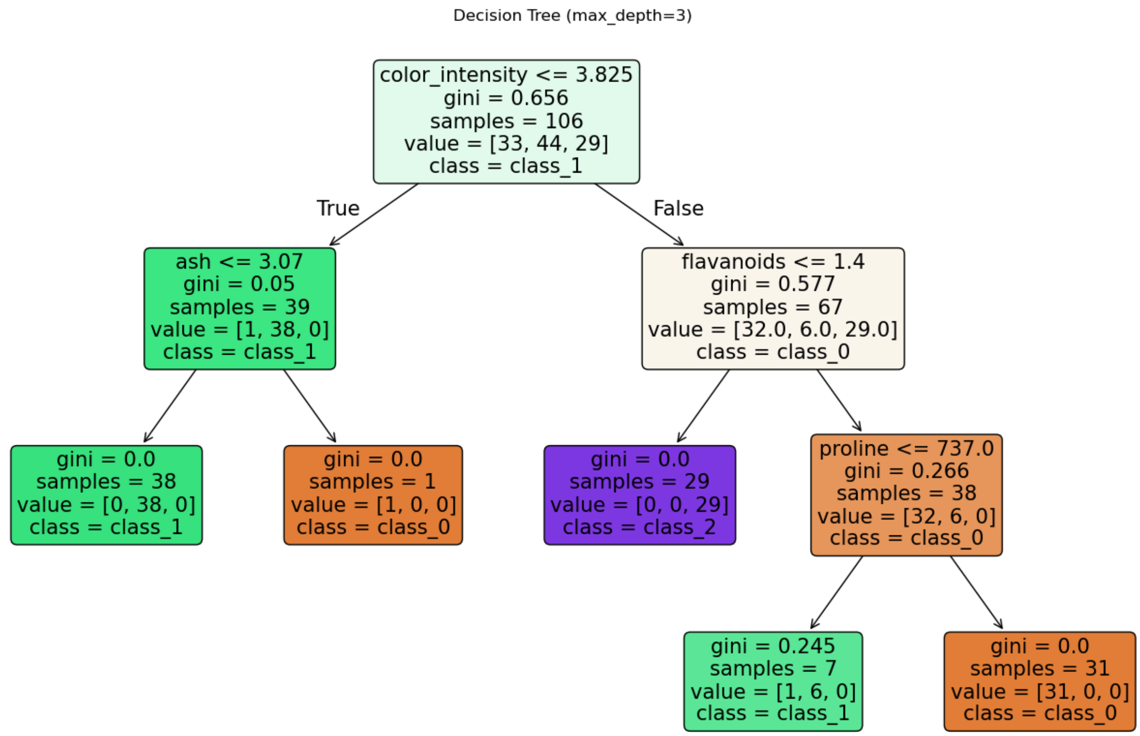
In [1]:
```
# Problem 1

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, c

# Load Wine dataset
data = load_wine()
X = data.data
y = data.target
feature_names = data.feature_names

# 1(a) PCA - First 2 principal components
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# 1(b) Explained variance ratio
explained_variance = pca.explained_variance_ratio_
print("Explained variance ratio of first two components:", explained_varianc

# 1(c) SVM on PCA components
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_s

svm = SVC()
svm.fit(X_train_pca, y_train)
y_pred_svm = svm.predict(X_test_pca)

# 1(d) Decision tree on original features
X_train, X_test, _, _ = train_test_split(X, y, test_size=0.4, random_state=4
tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)

# Metrics
print("\nSVM Classifier Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Precision (macro):", precision_score(y_test, y_pred_svm, average='mac
print("Recall (macro):", recall_score(y_test, y_pred_svm, average='macro'))

print("\nDecision Tree Classifier Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print("Precision (macro):", precision_score(y_test, y_pred_tree, average='ma
print("Recall (macro):", recall_score(y_test, y_pred_tree, average='macro'))

# 1(e) Plot the decision tree
plt.figure(figsize=(16, 10))
plot_tree(tree, feature_names=feature_names, class_names=data.target_names,
```

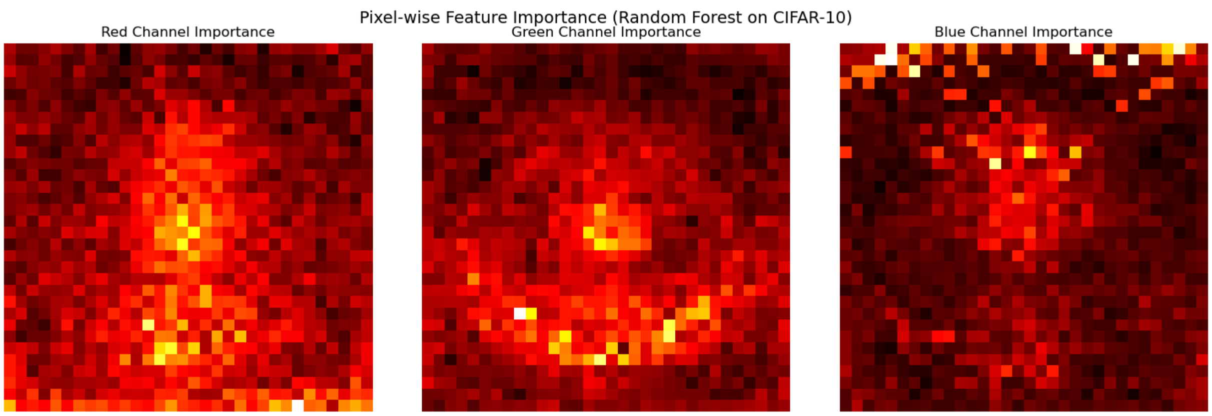
```
plt.title("Decision Tree (max_depth=3)")
plt.show()
```

```
Explained variance ratio of first two components: [0.99809123 0.00173592]

SVM Classifier Metrics:
Accuracy: 0.7083333333333334
Precision (macro): 0.6882332643202208
Recall (macro): 0.6934822812015794

Decision Tree Classifier Metrics:
Accuracy: 0.9305555555555556
Precision (macro): 0.9479166666666666
Recall (macro): 0.9311740890688259
```



In [2]:
```
#Problem 2
# Problem 2: Feature Importance on CIFAR-10 using Random Forest

from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.datasets import cifar10

# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
y_train = y_train.ravel()
y_test = y_test.ravel()

# Combine train and test sets
X = np.concatenate((x_train, x_test), axis=0)
Y = np.concatenate((y_train, y_test), axis=0)

# Flatten the images (32x32x3 -> 3072 features)
X_flat = X.reshape(X.shape[0], -1)
```

```python
# Train a Random Forest Classifier
rf = RandomForestClassifier(n_estimators=50, random_state=42, n_jobs=-1)
rf.fit(X_flat, Y)

# Extract feature importances and reshape to image dimensions
importances = rf.feature_importances_
importances_image = importances.reshape(32, 32, 3)

# Plot feature importance heatmaps per channel
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
channels = ['Red', 'Green', 'Blue']
for i in range(3):
    axs[i].imshow(importances_image[:, :, i], cmap='hot')
    axs[i].set_title(f'{channels[i]} Channel Importance')
    axs[i].axis('off')

plt.suptitle("Pixel-wise Feature Importance (Random Forest on CIFAR-10)", fc
plt.tight_layout()
plt.show()
```



In [6]:

```python
# Problem 3

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler

# Load temperature data
temp_data = surface_temp.reshape(-1, 1)

# Normalize data
scaler = MinMaxScaler()
temp_scaled = scaler.fit_transform(temp_data)

# Prepare data for RNN
def create_sequences(data, window_size, pred_size):
    X, y = [], []
    for i in range(len(data) - window_size - pred_size + 1):
        X.append(data[i:i+window_size])
        y.append(data[i+window_size:i+window_size+pred_size])
    return np.array(X), np.array(y)

window_size = 20  # how many past steps to look at
pred_size = 10    # predict next 10 steps
```

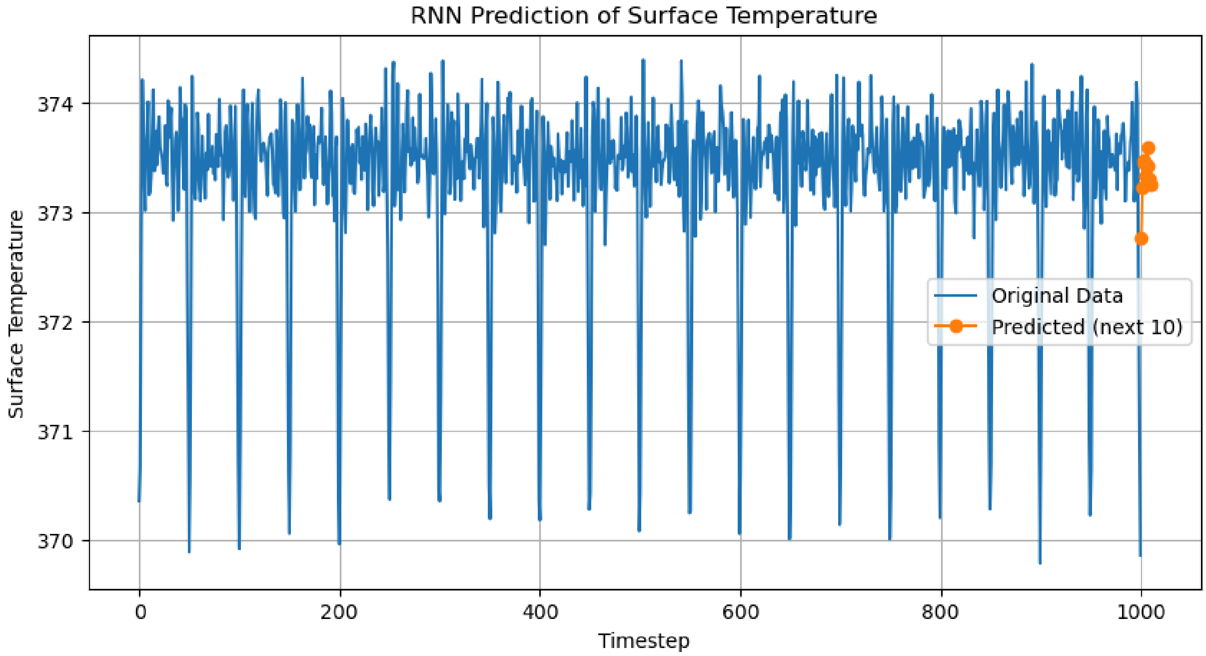
```
X, y = create_sequences(temp_scaled, window_size, pred_size)

# Split into training and validation sets
split = int(0.8 * len(X))
X_train, y_train = X[:split], y[:split]
X_val, y_val = X[split:], y[split:]

# Build the RNN model
model = Sequential([
    LSTM(64, activation='tanh', input_shape=(window_size, 1)),
    Dense(pred_size)
])
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=30, validation_data=(X_val, y_v

# Predict the next 10 timesteps using the last available window
last_window = temp_scaled[-window_size:].reshape(1, window_size, 1)
predicted_scaled = model.predict(last_window)

# Inverse transform predictions
predicted = scaler.inverse_transform(predicted_scaled.reshape(-1, 1)).flatte

# Plot the results
plt.figure(figsize=(10, 5))
plt.plot(range(1000), temp_data.flatten(), label='Original Data')
plt.plot(range(1000, 1010), predicted, label='Predicted (next 10)', marker='
plt.xlabel('Timestep')
plt.ylabel('Surface Temperature')
plt.title('RNN Prediction of Surface Temperature')
plt.legend()
plt.grid(True)
plt.show()
```

```
Epoch 1/30
```

```
C:\Users\Tom\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: Us
erWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
n using Sequential models, prefer using an `Input(shape)` object as the firs
t layer in the model instead.
  super().__init__(**kwargs)
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 7s 49ms/step - loss: 0.4678 - val_loss: 0.0338
Epoch 2/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0397 - val_loss: 0.0299
Epoch 3/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0328 - val_loss: 0.0283
Epoch 4/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0324 - val_loss: 0.0283
Epoch 5/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0300 - val_loss: 0.0278
Epoch 6/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0298 - val_loss: 0.0280
Epoch 7/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0322 - val_loss: 0.0277
Epoch 8/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.0316 - val_loss: 0.0273
Epoch 9/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0325 - val_loss: 0.0267
Epoch 10/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.0312 - val_loss: 0.0277
Epoch 11/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0321 - val_loss: 0.0266
Epoch 12/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0299 - val_loss: 0.0264
Epoch 13/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0291 - val_loss: 0.0265
Epoch 14/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0288 - val_loss: 0.0268
Epoch 15/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0278 - val_loss: 0.0262
Epoch 16/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.0280 - val_loss: 0.0268
Epoch 17/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0284 - val_loss: 0.0264
Epoch 18/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0279 - val_loss: 0.0262
Epoch 19/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0293 - val_loss: 0.0262
Epoch 20/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0278 - val_loss: 0.0259
Epoch 21/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0288 - val_loss: 0.0259
Epoch 22/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0284 - val_loss: 0.0259
Epoch 23/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0293 - val_loss: 0.0259
Epoch 24/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0300 - val_loss: 0.0258
Epoch 25/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0299 - val_loss: 0.0257
Epoch 26/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0276 - val_loss: 0.0256
Epoch 27/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0268 - val_loss: 0.0261
Epoch 28/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0291 - val_loss: 0.0257
Epoch 29/30
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0265 - val_loss: 0.0256
Epoch 30/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0296 - val_loss: 0.0255
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 441ms/step
```



In [ ]: