```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
class_names = iris.target_names

# Convert to DataFrame for easier data manipulation
df = pd.DataFrame(data=np.c_[X, y], columns=feature_names + ['species'])
df['species'] = df['species'].map({0: class_names[0], 1: class_names[1], 2: class_names[2]})

# Visualize data
sns.pairplot(df, hue='species')
plt.show()

# Split data 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)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Logistic Regression model
log_reg_model = LogisticRegression(random_state=42)
log_reg_model.fit(X_train, y_train)
y_pred_log_reg = log_reg_model.predict(X_test)
log_reg_acc = accuracy_score(y_test, y_pred_log_reg)
```

```
print("Logistic Regression Accuracy:", log_reg_acc)
print(classification_report(y_test, y_pred_log_reg))

# Support Vector Machine model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
svm_acc = accuracy_score(y_test, y_pred_svm)

print("Support Vector Machine Accuracy:", svm_acc)
print(classification_report(y_test, y_pred_svm))

# Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
rf_acc = accuracy_score(y_test, y_pred_rf)

print("Random Forest Accuracy:", rf_acc)
print(classification_report(y_test, y_pred_rf))

# Neural Network model using TensorFlow
nn_model = Sequential([
    Dense(16, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(16, activation='relu'),
    Dense(3, activation='softmax')
])

# Compile the model
nn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Train the neural network model
nn_model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Predict on the test set
y_pred_nn = nn_model.predict(X_test)
y_pred_nn_classes = np.argmax(y_pred_nn, axis=1)
nn_acc = accuracy_score(y_test, y_pred_nn_classes)

print("Neural Network Accuracy:", nn_acc)
print(classification_report(y_test, y_pred_nn_classes))

# Plot confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred_nn_classes)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names,
yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Choose the model with the best performance based on accuracy and classification report

# Model deployment
best_model = nn_model if nn_acc > max(log_reg_acc, svm_acc, rf_acc) else log_reg_model if
log_reg_acc > max(svm_acc, rf_acc) else svm_model if svm_acc > rf_acc else rf_model

# Use the best model for classifying new iris flowers
new_data = np.array([[5.1, 3.5, 1.4, 0.2]])  # Replace with new data
new_data_scaled = scaler.transform(new_data)
predicted_class = best_model.predict(new_data_scaled)
predicted_class_name = class_names[int(predicted_class)]

print(f"Predicted Class for the new data: {predicted_class_name}")
```