Practical Experiment: Implementing Time Series Models

Objective:

To implement various time series models on the UCI Chess dataset using TensorFlow.

Steps:

1. **Read and Preprocess Data**:
   - Use the Pandas library to read the dataset.
   - Preprocess the data for time series analysis.
2. **Implement ARIMA Model**:
   - Use the statsmodels library to implement the ARIMA model.
3. **Implement LSTM Model**:
   - Use TensorFlow to implement the LSTM model.
4. **Implement Prophet Model**:
   - Use the Facebook Prophet library to implement the Prophet model.

Data File:

The UCI Chess dataset can be found here.

Python Script:

python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.arima.model import ARIMA
from fbprophet import Prophet

# Step 1: Read and Preprocess Data
url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/chess/kr-vs-kp.data'
data = pd.read_csv(url, header=None)

# Assuming the data has a time component, for example:
data['date'] = pd.date_range(start='1/1/2000', periods=len(data),
freq='D')

# Convert data to time series
data.set_index('date', inplace=True)

# For demonstration, let's create a synthetic time series column
data['value'] = np.sin(np.linspace(0, 3.14, len(data)))
```

```
# Plot the time series
plt.figure(figsize=(10, 6))
plt.plot(data['value'])
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.show()

# Step 2: Implement ARIMA Model
arima_model = ARIMA(data['value'], order=(5, 1, 0))
arima_result = arima_model.fit()
print(arima_result.summary())

# Forecast with ARIMA
arima_forecast = arima_result.forecast(steps=30)
plt.figure(figsize=(10, 6))
plt.plot(data['value'], label='Original')
plt.plot(pd.date_range(start=data.index[-1], periods=30, freq='D'),
arima_forecast, label='Forecast')
plt.title('ARIMA Forecast')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()

# Step 3: Implement LSTM Model
# Scaling the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data['value'].values.reshape(-1,
1))

# Creating a dataset with time steps
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

time_step = 10
X, Y = create_dataset(scaled_data, time_step)

# Reshaping the input to be [samples, time steps, features]
X = X.reshape(X.shape[0], X.shape[1], 1)

# Creating the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step,
1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# Compile the model
```

```
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X, Y, batch_size=1, epochs=1)

# Forecast with LSTM
train_data = scaled_data[:len(data) - 30]
test_data = scaled_data[len(data) - 30 - time_step:]

X_test, Y_test = create_dataset(test_data, time_step)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)

# Plot LSTM forecast
plt.figure(figsize=(10, 6))
plt.plot(data.index, data['value'], label='Original')
plt.plot(data.index[-len(predictions):], predictions,
label='Forecast')
plt.title('LSTM Forecast')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()

# Step 4: Implement Prophet Model
df = data.reset_index().rename(columns={'date': 'ds', 'value': 'y'})
prophet_model = Prophet()
prophet_model.fit(df)

# Forecast with Prophet
future = prophet_model.make_future_dataframe(periods=30)
forecast = prophet_model.predict(future)

# Plot Prophet forecast
fig = prophet_model.plot(forecast)
plt.title('Prophet Forecast')
plt.xlabel('Date')
plt.ylabel('Value')
plt.show()
```

Explanation:

1. **Read and Preprocess Data**:
    - Load the dataset and create a synthetic time series column for demonstration.
    - Plot the time series data.
2. **Implement ARIMA Model**:
    - Use the statsmodels library to create and fit the ARIMA model.
    - Forecast future values and plot the forecast.
3. **Implement LSTM Model**:
    - Scale the data using MinMaxScaler.

- Create a dataset with time steps for the LSTM model.
- Build and train the LSTM model using TensorFlow.
- Forecast future values and plot the forecast.

4. **Implement Prophet Model**:
    - Use the Facebook Prophet library to create and fit the Prophet model.
    - Forecast future values and plot the forecast.