[Add your name here] Assignment #4 Time Series Analysis

September 27, 2023

[ ]:

```
# import libraries
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# nothing will go to output
```

[ ]:

```
# Generate synthetic time series data
np.random.seed(0)
n_samples = 100
X = np.arange(n_samples).reshape(-1, 1)
y = 3 * X + 10 + np.random.randn(n_samples, 1)

# sanity check
print("X (Features):")
print(X[:5])  # Print the first 5 rows of X
print("\ny (Labels):")
print(y[:5])  # Print the first 5 rows of y
```

[ ]:

```
# plot what we have at this point
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(X, y, label="Synthetic Data")
plt.xlabel("Time")
plt.ylabel("Value")
plt.legend()
plt.show()
```

[ ]:

```
# summanry statistics to verify that data has expected characteristics
print("Mean of X:", np.mean(X))
print("Standard Deviation of X:", np.std(X))
print("Mean of y:", np.mean(y))
print("Standard Deviation of y:", np.std(y))
```

[ ]:

```
# Split the data into training and testing sets
split_ratio = 0.8
split_index = int(split_ratio * n_samples)
X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]

#
# Print the sizes (number of samples) of the training and testing
# sets to verify that the split proportions are as expected
#

print(f"Training set size: {len(X_train)} samples")
print(f"Testing set size: {len(X_test)} samples")
```

[ ]:

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

# sanity check, Print the first 5 rows
print(X_test_scaled[:5])
```

[ ]:

```
# Train a linear regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train) # learns the coefficients (slope and intercept) that best fit training data


# Extract the slope (coefficient) and intercept of the model
slope = model.coef_[0][0]
intercept = model.intercept_[0]

# Create a scatter plot of the data points
plt.scatter(X, y, label="Data Points")

# Create the regression line using the slope and intercept
regression_line = slope * X + intercept

# Plot the regression line
plt.plot(X, regression_line, color='red', label="Regression Line")

plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.title(f"Linear Regression\nSlope: {slope:.2f}, Intercept: {intercept:.2f}")
plt.show()
```