**Great! As a beginner, starting with a cryptocurrency price predictor might be a bit easier due to simpler data structures and more readily available sentiment data like the Fear & Greed Index. Here’s a step-by-step guide to help you get started:**

**### Step-by-Step Guide to Building a Cryptocurrency Price Predictor**

**#### 1. \*\*Set Up Your Environment:\*\***

**- Install Python if you haven't already.**

**- Install necessary libraries: `pandas`, `numpy`, `matplotlib`, `sklearn`, `keras`, `tensorflow`, `requests`, and `bs4` (for web scraping).**

**```bash**

**pip install pandas numpy matplotlib scikit-learn keras tensorflow requests beautifulsoup4**

**```**

**#### 2. \*\*Collect Data:\*\***

**- Use an API to get historical cryptocurrency data (e.g., Bitcoin) and sentiment data (Fear & Greed Index).**

**##### Example: Fetching Bitcoin Data from CoinGecko**

**```python**

**import requests**

**import pandas as pd**

**# Fetch historical Bitcoin data**

**url = 'https://api.coingecko.com/api/v3/coins/bitcoin/market\_chart'**

**params = {'vs\_currency': 'usd', 'days': 'max'}**

**response = requests.get(url, params=params)**

**data = response.json()**

**# Convert to DataFrame**

**prices = data['prices']**

**df = pd.DataFrame(prices, columns=['timestamp', 'price'])**

**df['date'] = pd.to\_datetime(df['timestamp'], unit='ms')**

**df.set\_index('date', inplace=True)**

**df.drop('timestamp', axis=1, inplace=True)**

**# Save to CSV**

**df.to\_csv('bitcoin\_prices.csv')**

**```**

**##### Example: Fetching Fear & Greed Index**

**```python**

**url = 'https://api.alternative.me/fng/?limit=0'**

**response = requests.get(url)**

**data = response.json()**

**# Convert to DataFrame**

**fng\_data = pd.DataFrame(data['data'])**

**fng\_data['timestamp'] = pd.to\_datetime(fng\_data['timestamp'], unit='s')**

**fng\_data.set\_index('timestamp', inplace=True)**

**fng\_data.to\_csv('fear\_greed\_index.csv')**

**```**

**#### 3. \*\*Load and Merge Data:\*\***

**```python**

**# Load the datasets**

**bitcoin\_data = pd.read\_csv('bitcoin\_prices.csv', parse\_dates=['date'], index\_col='date')**

**fng\_data = pd.read\_csv('fear\_greed\_index.csv', parse\_dates=['timestamp'], index\_col='timestamp')**

**# Merge datasets on date**

**merged\_data = bitcoin\_data.join(fng\_data, how='inner')**

**# Save merged data to CSV**

**merged\_data.to\_csv('merged\_data.csv')**

**```**

**#### 4. \*\*Preprocess Data:\*\***

**```python**

**import numpy as np**

**# Fill missing values if any**

**merged\_data.fillna(method='ffill', inplace=True)**

**# Normalize data**

**from sklearn.preprocessing import MinMaxScaler**

**scaler = MinMaxScaler(feature\_range=(0, 1))**

**scaled\_data = scaler.fit\_transform(merged\_data)**

**# Convert back to DataFrame**

**scaled\_df = pd.DataFrame(scaled\_data, columns=merged\_data.columns, index=merged\_data.index)**

**```**

**#### 5. \*\*Create Features and Labels:\*\***

**```python**

**def create\_dataset(data, time\_step=1):**

**X, Y = [], []**

**for i in range(len(data) - time\_step - 1):**

**a = data[i:(i + time\_step), :]**

**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)**

**# Split into training and testing data**

**train\_size = int(len(X) \* 0.8)**

**test\_size = len(X) - train\_size**

**X\_train, X\_test = X[:train\_size], X[train\_size:]**

**Y\_train, Y\_test = Y[:train\_size], Y[train\_size:]**

**```**

**#### 6. \*\*Build and Train the Model:\*\***

**```python**

**from keras.models import Sequential**

**from keras.layers import Dense, LSTM**

**# Create the LSTM model**

**model = Sequential()**

**model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, scaled\_data.shape[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\_train, Y\_train, batch\_size=1, epochs=1)**

**```**

**#### 7. \*\*Evaluate the Model:\*\***

**```python**

**# Predictions**

**train\_predict = model.predict(X\_train)**

**test\_predict = model.predict(X\_test)**

**# Inverse transform to get actual values**

**train\_predict = scaler.inverse\_transform(np.concatenate((train\_predict, np.zeros((train\_predict.shape[0], scaled\_data.shape[1]-1))), axis=1))[:,0]**

**test\_predict = scaler.inverse\_transform(np.concatenate((test\_predict, np.zeros((test\_predict.shape[0], scaled\_data.shape[1]-1))), axis=1))[:,0]**

**# Calculate RMSE**

**import math**

**from sklearn.metrics import mean\_squared\_error**

**train\_rmse = math.sqrt(mean\_squared\_error(Y\_train, train\_predict))**

**test\_rmse = math.sqrt(mean\_squared\_error(Y\_test, test\_predict))**

**print(f'Train RMSE: {train\_rmse}')**

**print(f'Test RMSE: {test\_rmse}')**

**```**

**#### 8. \*\*Visualize the Results:\*\***

**```python**

**import matplotlib.pyplot as plt**

**# Plotting**

**plt.figure(figsize=(16,8))**

**plt.plot(merged\_data.index[time\_step+1:len(train\_predict)+time\_step+1], train\_predict, label='Train Predict')**

**plt.plot(merged\_data.index[len(train\_predict)+(time\_step\*2)+1:], test\_predict, label='Test Predict')**

**plt.plot(merged\_data.index, merged\_data['price'], label='Actual Price')**

**plt.legend()**

**plt.show()**

**```**

**This should give you a basic cryptocurrency price predictor model. You can further refine and improve it as you gain more experience. Let me know if you need any more help or details on any of these steps!**