Documentation for Newcomers: Bitcoin Price Prediction using Support Vector Regression (SVR)

This Python program uses **Support Vector Regression (SVR)** to predict future Bitcoin prices based on historical data. The key steps of the program involve reading a dataset, preprocessing the data, training a model, testing the model, and making predictions. Here's a breakdown of the code and its components:

---

1. Importing Libraries

python

Copy code

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.svm import SVR
```

- **numpy**: A library used for numerical operations and handling arrays.
- **pandas**: A library for data manipulation and analysis (works with DataFrames).
- **train_test_split**: A function from **scikit-learn** that splits the data into training and testing sets.
- **SVR**: Support Vector Regression model from **scikit-learn**, used for regression tasks.

---

2. Reading the Dataset

python

Copy code

```
df = pd.read_csv('bitcoin.csv')

df.head()
```

- **pd.read_csv('bitcoin.csv')**: Reads a CSV file containing Bitcoin data (such as dates and prices).
- **df.head()**: Displays the first 5 rows of the dataset to check the data.

---

3. Data Preprocessing

- **Dropping the 'Date' column**:

python

Copy code

```
df.drop(['Date'], axis=1, inplace=True)
```

  - This removes the **'Date'** column from the DataFrame, as it's not needed for the prediction model.

- **Creating the Prediction Column**:

python

Copy code

```
df['Prediction'] = df[['Price']].shift(-predictionDays)
```

  - Adds a new column called **'Prediction'**, which shifts the 'Price' column by predictionDays (30 in this case), essentially showing the future Bitcoin price for the next 30 days.

---

4. Creating the Independent and Dependent Datasets

- **Independent Data (Features)**:

python

Copy code

```
x = np.array(df.drop(['Prediction'], axis=1))
```

  - **x** is the independent dataset (features), which includes all columns except the **'Prediction'** column.

- **Dependent Data (Target)**:

python

Copy code

```
y = np.array(df['Prediction'])
```

  - **y** is the dependent dataset (target), which contains the future Bitcoin prices from the **'Prediction'** column.

- **Removing the last 30 rows from x and y**:

python

Copy code

```
x = x[:len(df)-predictionDays]
y = y[:-predictionDays]
```

- Since the last predictionDays rows don't have a corresponding target value (future prices), we remove them.

---

5. Splitting the Data into Training and Testing Sets

python

Copy code

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)
```

- **train_test_split** splits the data into:
  - **xtrain, ytrain`**: Data used to train the model.
  - **xtest, ytest`**: Data used to test the model.
- **test_size=0.2**: 20% of the data is used for testing, and 80% is used for training.

---

6. Training the Model with SVR

python

Copy code

```
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.00001)
```

- **SVR**: Creates an instance of the Support Vector Regression model.
  - **kernel='rbf'**: Specifies the Radial Basis Function kernel, which is often used in non-linear regression tasks.
  - **C=1e3**: Regularization parameter that controls overfitting.
  - **gamma=0.00001**: Defines how much influence a single training point has; smaller values create a smoother model.

python

Copy code

```
svr_rbf.fit(xtrain, ytrain)
```

- Trains the SVR model using the **training data (xtrain and ytrain)**.

---

7. Model Evaluation and Predictions

- **Evaluating the Model**:

python

Copy code

```
svr_rbf_confidence = svr_rbf.score(xtest, ytest)
print('SVR_RBF accuracy :', svr_rbf_confidence)
```

  - **svr_rbf.score()**: Measures the accuracy of the model on the test set. It outputs the **R^2 score**, a metric for how well the model predicts.

- **Making Predictions**:

python

Copy code

```
svm_prediction = svr_rbf.predict(xtest)
print(svm_prediction)
print()
print(ytest)
```

  - **svr_rbf.predict(xtest)**: Predicts the Bitcoin prices for the test data (xtest).
  - **ytest**: The actual test values for comparison.

---

8. Making Predictions for the Next 30 Days

python

Copy code

```
predictionDays_array = np.array(df.drop(['Prediction'], axis=1))[-predictionDays:]
```

```
svm_prediction = svr_rbf.predict(predictionDays_array)

print(svm_prediction)
```

- **predictionDays_array**: Takes the last 30 rows from the original dataset (excluding the **'Prediction'** column).
- **svm_prediction**: Predicts Bitcoin prices for the next 30 days based on this data.

---

9. Displaying the Actual Prices for the Last 30 Days

python

Copy code

```
print(df.tail(predictionDays))
```

- **df.tail(predictionDays)**: Displays the last 30 rows of the dataset, showing the actual Bitcoin prices for comparison.

Key Points to Understand:

- **SVR (Support Vector Regression)**: A machine learning algorithm used for predicting continuous values (like Bitcoin prices).
- **Data Preprocessing**: Important steps like dropping unnecessary columns and creating shifted columns for prediction.
- **Train-Test Split**: Dividing the dataset into training and testing sets to evaluate model performance.
- **Model Training and Testing**: Using svr_rbf.fit() to train the model and svr_rbf.predict() to make predictions.
- **Accuracy Evaluation**: The score() method is used to evaluate how well the model performs on unseen data (test data).