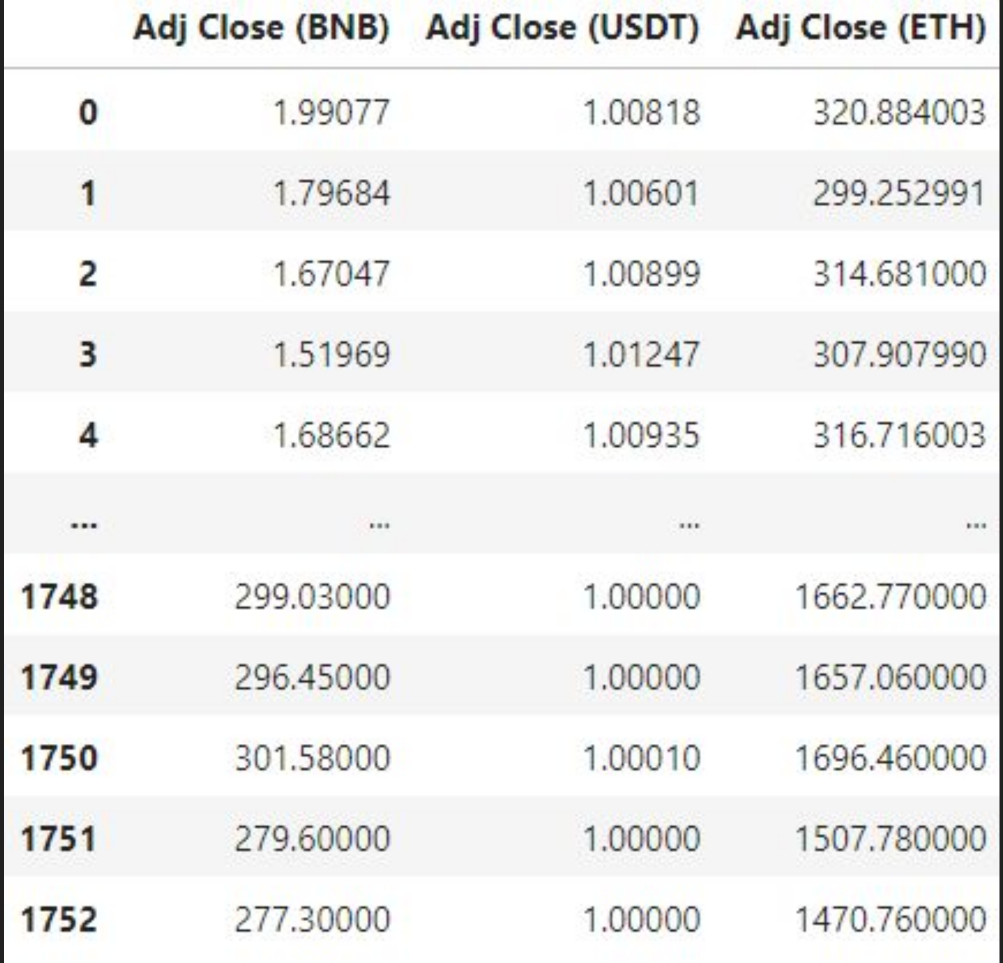
REGRESOR DE PRECIO DE CRIPTOMONEDAS

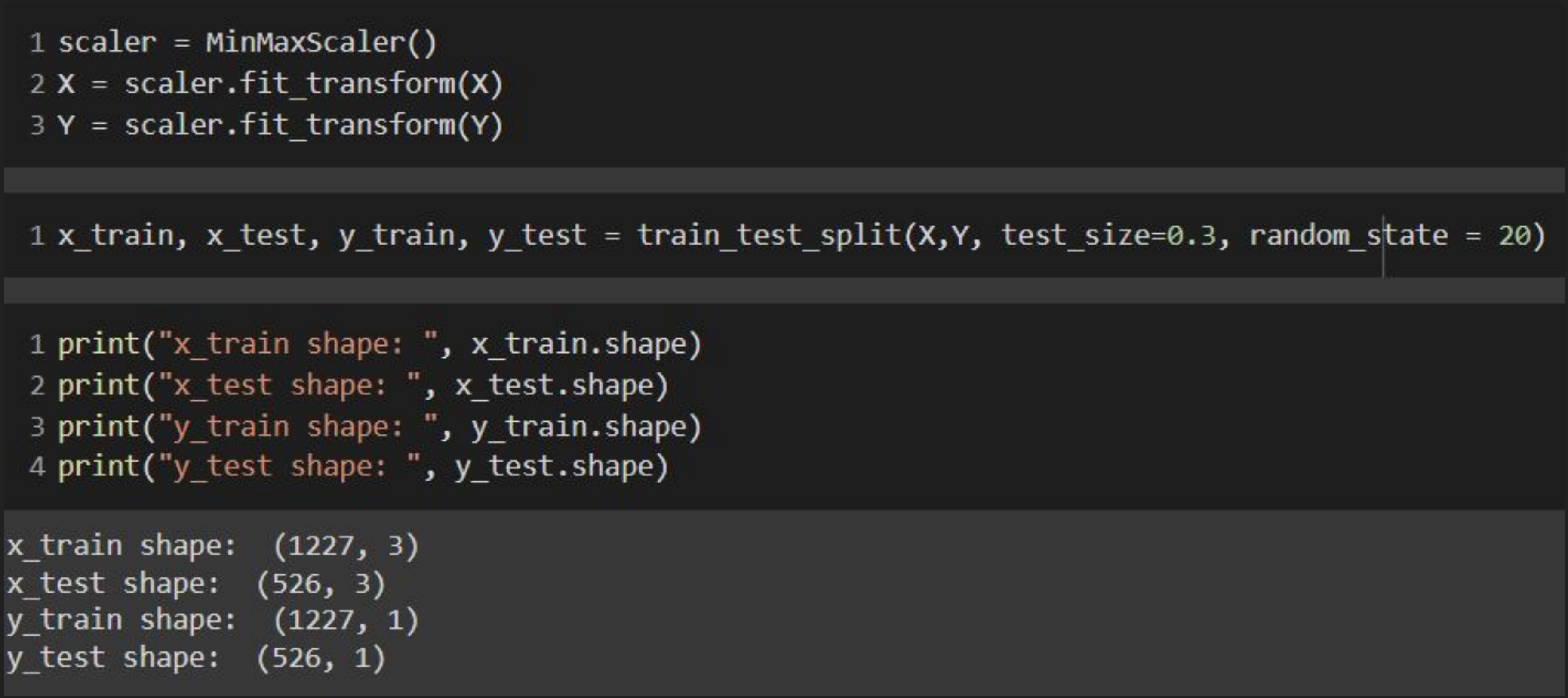
Jose Isaac Leal Manosalva
Jean Carlo Rodriguez Sanchez

DATASET
:



	Adj Close (BNB)	Adj Close (USDT)	Adj Close (ETH)
0	1.99077	1.00818	320.884003
1	1.79684	1.00601	299.252991
2	1.67047	1.00899	314.681000
3	1.51969	1.01247	307.907990
4	1.68662	1.00935	316.716003
...	...	...	...
1748	299.03000	1.00000	1662.770000
1749	296.45000	1.00000	1657.060000
1750	301.58000	1.00010	1696.460000
1751	279.60000	1.00000	1507.780000
1752	277.30000	1.00000	1470.760000

Normalización del dataset y particiones de train y test



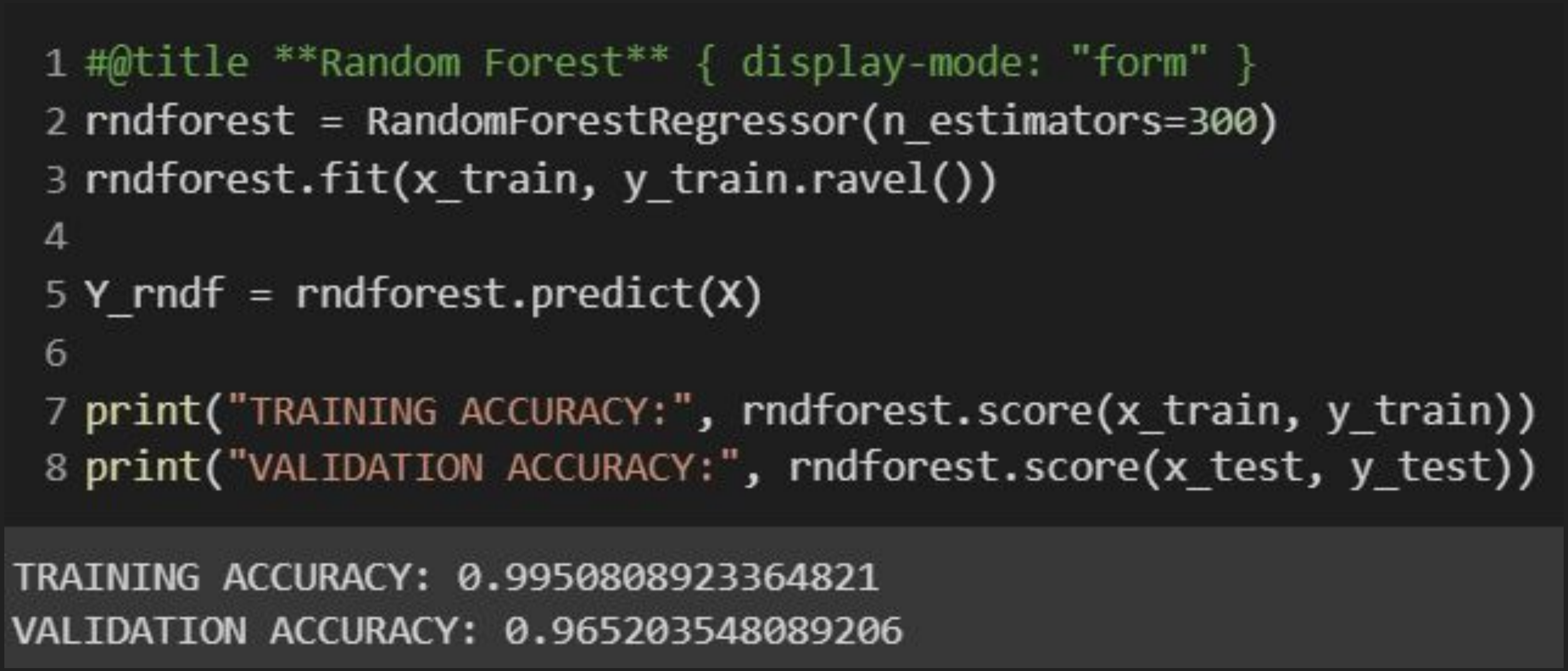
```python
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
Y = scaler.fit_transform(Y)
```

```python
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3, random_state = 20)
```

```python
print("x_train shape: ", x_train.shape)
print("x_test shape: ", x_test.shape)
print("y_train shape: ", y_train.shape)
print("y_test shape: ", y_test.shape)
```

```
x_train shape:  (1227, 3)
x_test shape:  (526, 3)
y_train shape:  (1227, 1)
y_test shape:  (526, 1)
```

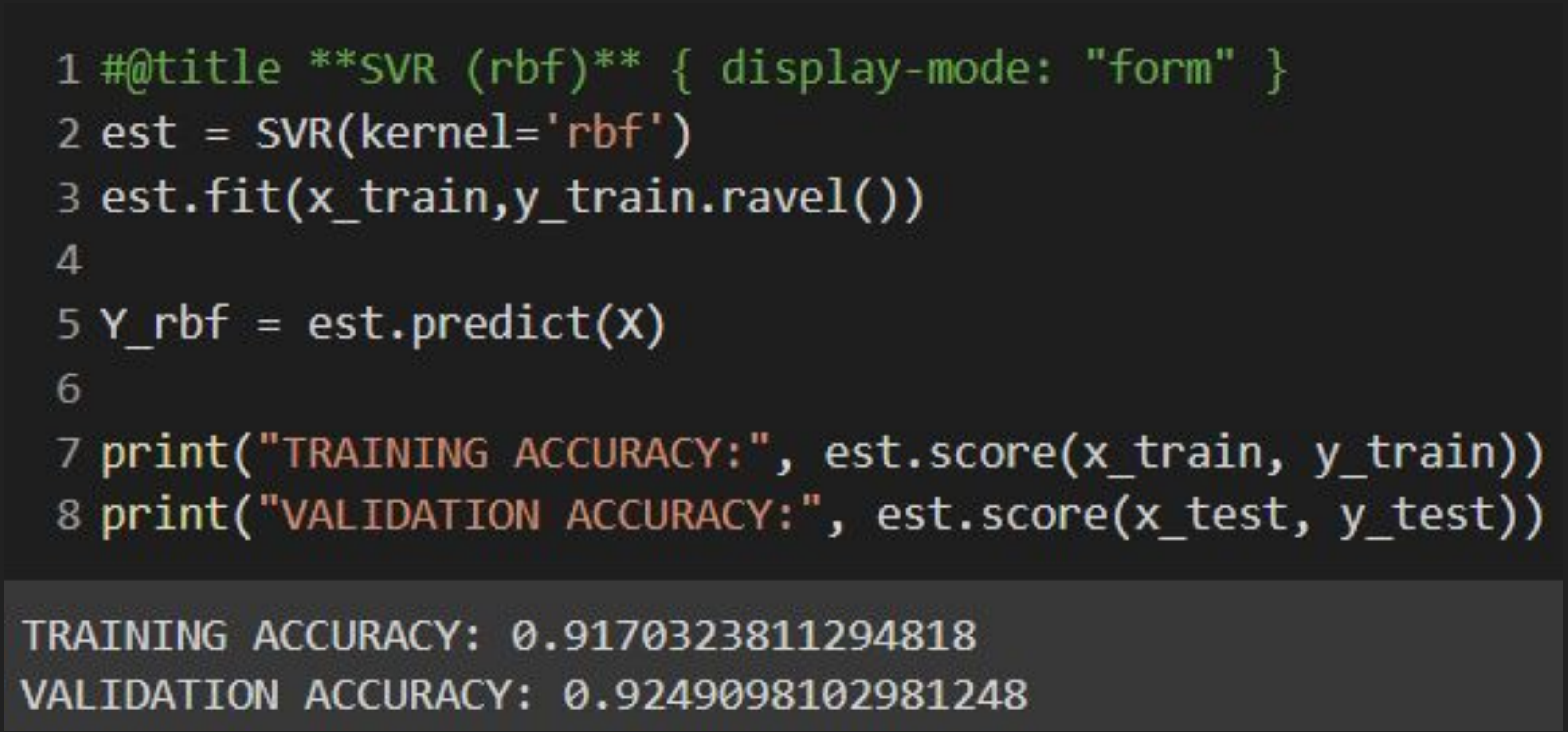
Random Forest utilizado y sus accuracy



```
#@title **Random Forest** { display-mode: "form" }
rndforest = RandomForestRegressor(n_estimators=300)
rndforest.fit(x_train, y_train.ravel())

Y_rndf = rndforest.predict(X)

print("TRAINING ACCURACY:", rndforest.score(x_train, y_train))
print("VALIDATION ACCURACY:", rndforest.score(x_test, y_test))
```

```
TRAINING ACCURACY: 0.9950808923364821
VALIDATION ACCURACY: 0.965203548089206
```

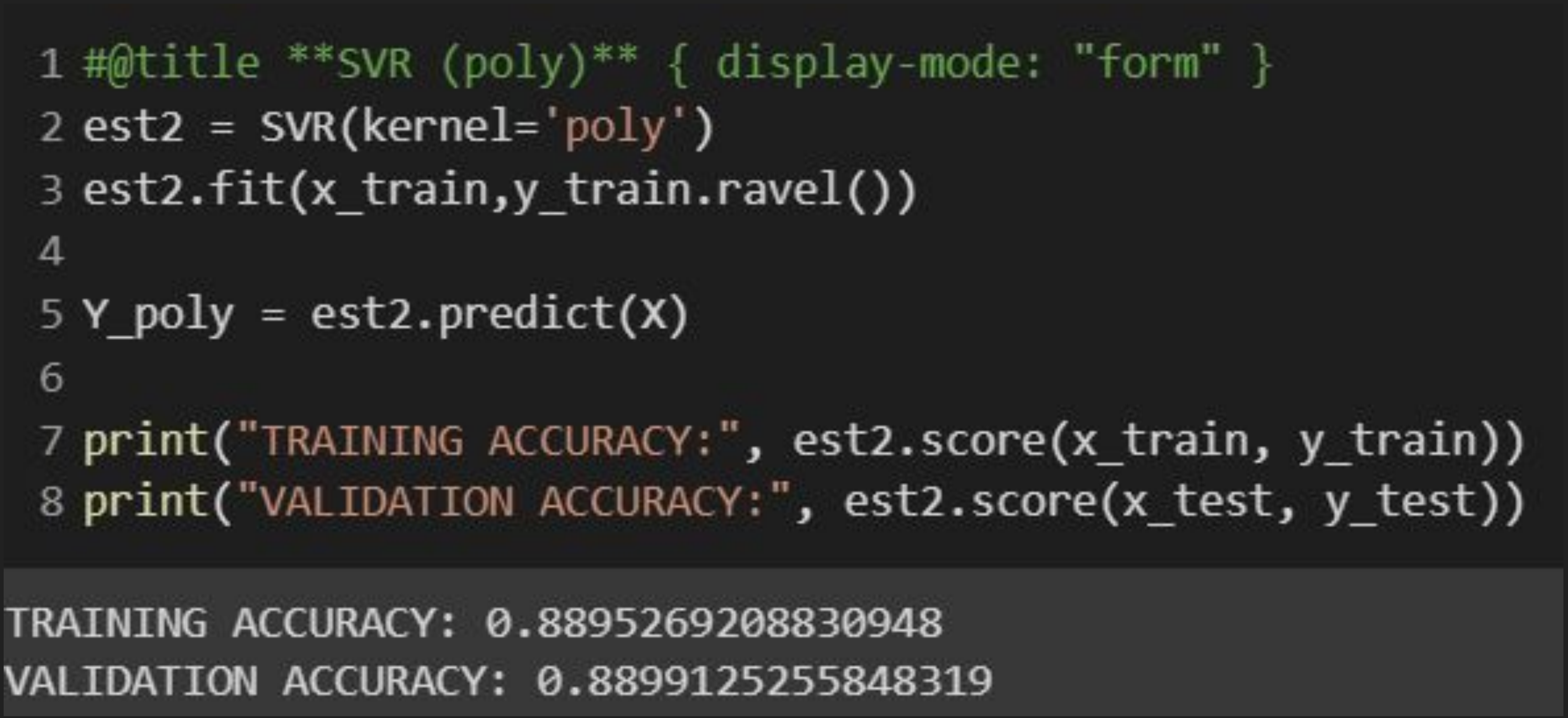
SVR con kernel rbf y su accuracy



```python
#@title **SVR (rbf)** { display-mode: "form" }
est = SVR(kernel='rbf')
est.fit(x_train,y_train.ravel())

Y_rbf = est.predict(X)

print("TRAINING ACCURACY:", est.score(x_train, y_train))
print("VALIDATION ACCURACY:", est.score(x_test, y_test))
```

```
TRAINING ACCURACY: 0.9170323811294818
VALIDATION ACCURACY: 0.9249098102981248
```

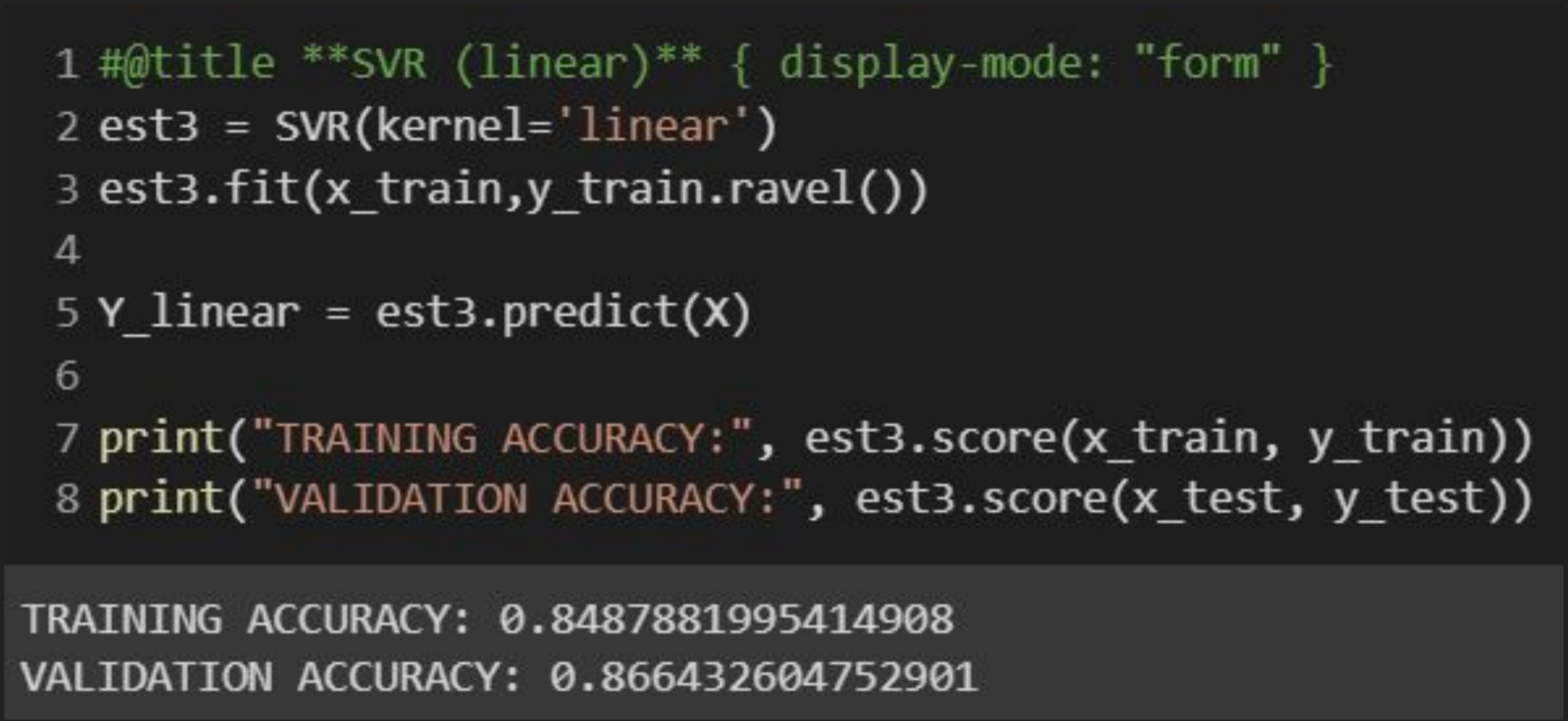
SVR con kernel Poly y sus accuracy



```python
#@title **SVR (poly)** { display-mode: "form" }
est2 = SVR(kernel='poly')
est2.fit(x_train,y_train.ravel())

Y_poly = est2.predict(X)

print("TRAINING ACCURACY:", est2.score(x_train, y_train))
print("VALIDATION ACCURACY:", est2.score(x_test, y_test))
```

```
TRAINING ACCURACY: 0.8895269208830948
VALIDATION ACCURACY: 0.8899125255848319
```

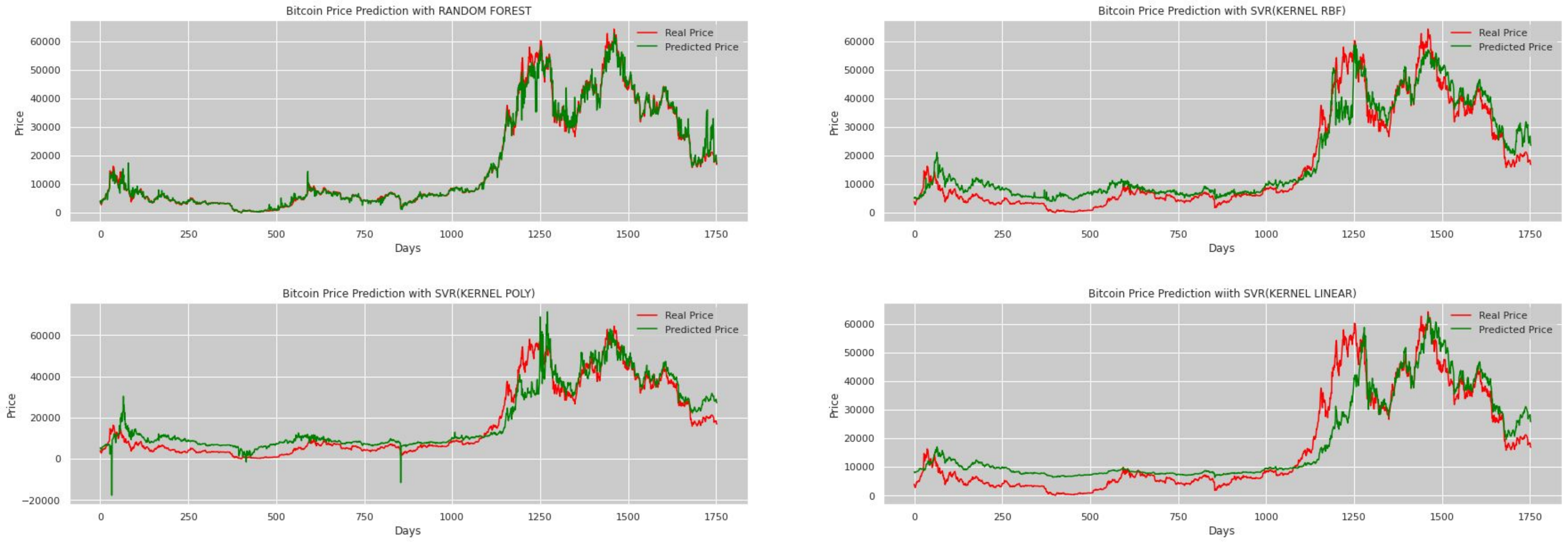
SVR con kernel linear y su accuracy



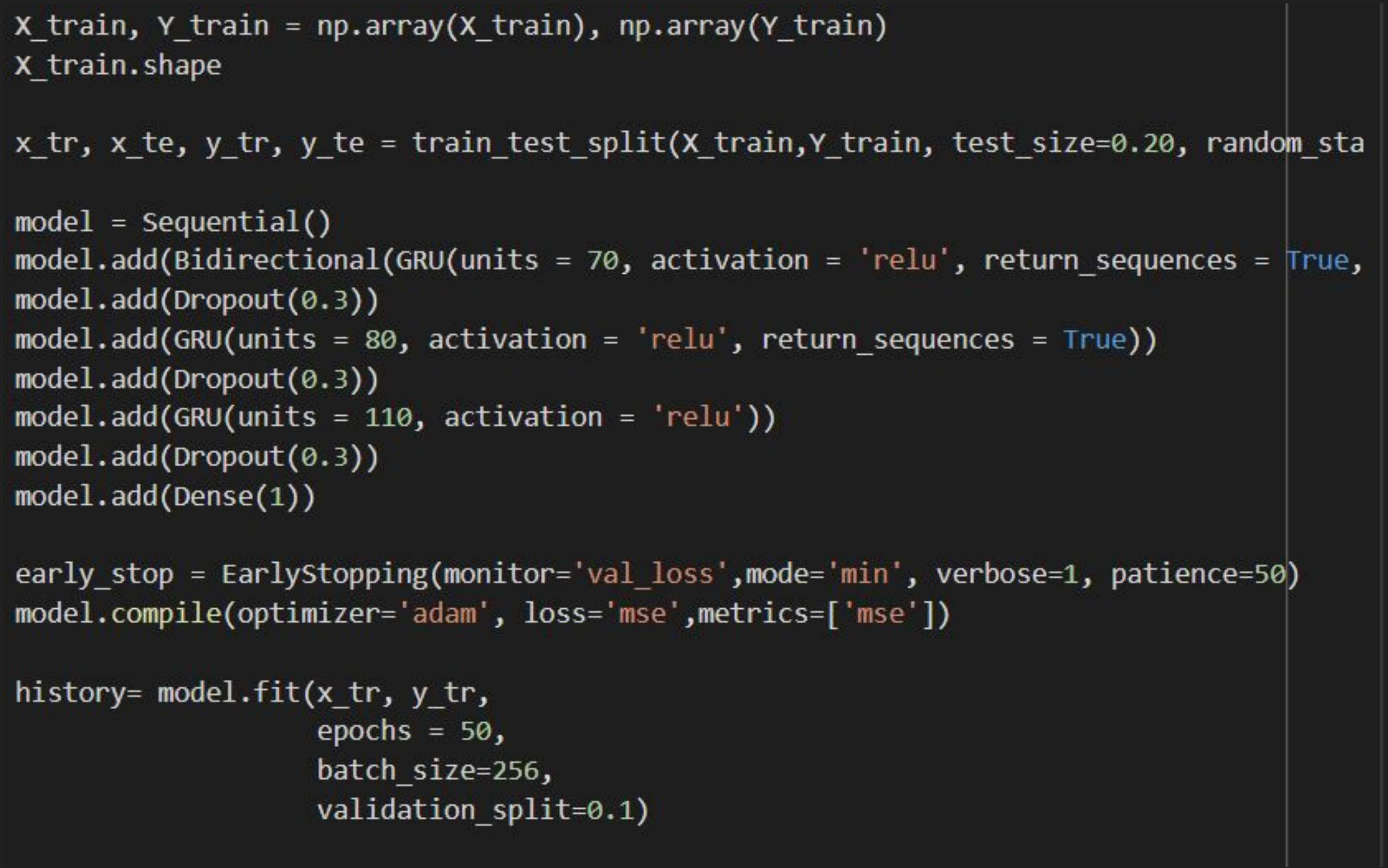
```
#@title **SVR (linear)** { display-mode: "form" }
est3 = SVR(kernel='linear')
est3.fit(x_train,y_train.ravel())

Y_linear = est3.predict(X)

print("TRAINING ACCURACY:", est3.score(x_train, y_train))
print("VALIDATION ACCURACY:", est3.score(x_test, y_test))
```

```
TRAINING ACCURACY: 0.8487881995414908
VALIDATION ACCURACY: 0.866432604752901
```

Resultados obtenidos

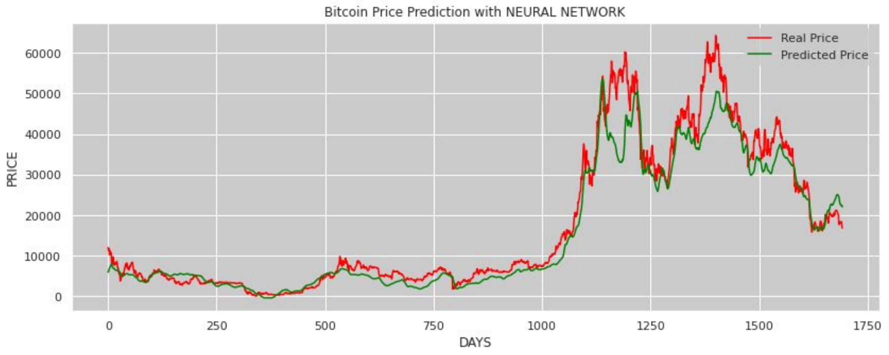


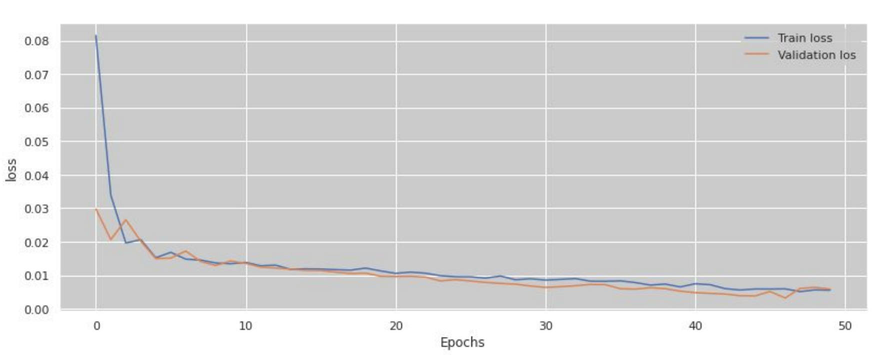
RNN utilizada



```
X_train, Y_train = np.array(X_train), np.array(Y_train)
X_train.shape

x_tr, x_te, y_tr, y_te = train_test_split(X_train,Y_train, test_size=0.20, random_sta

model = Sequential()
model.add(Bidirectional(GRU(units = 70, activation = 'relu', return_sequences = True,
model.add(Dropout(0.3))
model.add(GRU(units = 80, activation = 'relu', return_sequences = True))
model.add(Dropout(0.3))
model.add(GRU(units = 110, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(1))

early_stop = EarlyStopping(monitor='val_loss',mode='min', verbose=1, patience=50)
model.compile(optimizer='adam', loss='mse',metrics=['mse'])

history= model.fit(x_tr, y_tr,
                   epochs = 50,
                   batch_size=256,
                   validation_split=0.1)
```

Resultados obtenidos con la RNN


Bitcoin Price Prediction with NEURAL NETWORK
Real Price
Predicted Price
PRICE
DAYS


Train loss
Validation los
loss
Epochs