GRENFIN ASSIGNMENT 3

Greening Motion Groupwork

Commodity price discovery using natural variables: An application of statistical and Machine Learning models to Cocoa's future prices

Authors
Vittorio Balestrieri - Nuri Can Ozkan
Kian Ekramnosratian - Giancarlo Meléndez
Davide Previtali - Giovanni Cioli Puviani

April 29, 2023

Contents

1 Abstract

This project consists of an empirical analysis on cocoa future prices considering as drivers non-financial variables: cocoa is a commodity that is widely traded in financial markets, and its price can be influenced by various factors. Among those, changes in weather patterns can affect cocoa production and lead to supply disruptions, that can in turn greatly influence the price of cocoa.

Historical data concerning natural variables are collected and chosen to be specific to where production is located and with respect to cocoa farming idiosyncrasies. Those are analysed singularly together with the historical future prices of cocoa traded in the US.

Afterwards, quantitative models are implemented to evaluate their relationships to check if the selected variables can be used to predict the movement of future prices. This is useful to provide an indication for hedging climate risks in cocoa markets and to understand the dynamics of commodity prices with respect to weather-related risk.

2 Methodology and Data

Cocoa is a tropical crop, and its production is concentrated in few countries, primarily in West Africa. Ivory Coast and Nigeria account together for nearly 70 percent of global production of Cocoa (https://worldpopulationreview.com/country-rankings/cocoa-producing-countries).

Consequently, the production in these areas can influence the price of cocoa in the US, where high volumes of commodities are exchanged. Also, from the data of the International Cocoa Organization (https://www.icco.org/growing-cocoa/), we can observe that Temperature, Rainfalls and Humidity are the most relevant weather-related risk factors for Cocoa production. These considerations are the starting point of our analysis.

2.1 Data sources and variables

Historical monthly Data regarding the risk factors mentioned before are collected from the POWER DAVe (Prediction of Worldwide Energy Resource (POWER) — Data Access Viewer Enhanced (DAVe) - https://power.larc.nasa.gov/beta/data-access-viewer/) provided by NASA, selected in an area between Ivory Coast and Nigeria and detected with monthly meteorology. in particular:

- T2M: Mean Temperature at 2 meters expressed in Kelvin.
- PRECTOTCORR: Mean amount of Precipitations, expressed in kg · m-2 · s-1.
- QV2M: Mean Specific Humidity at 2 Meters, expressed in kg/kg
- T2MMAX and T2MMIN: Maximum and Minimum temperature among the stations, expressed in Kelvin

Variable	Mean	Dev std	Min	Max	N
PRECTOTCORR	3.4636779	2.2346377	0.0540000	10.6270000	444
QV2M	15.3123401	2.5769855	7.3100000	18.4500000	444
T2M	26.4216824	1.4538365	23.6490000	30.4020000	444
T2M_MAX	34.1535631	2.7820794	29.1080000	40.3610000	444
T2M_MIN	19.9820225	2.1257512	13.5310000	24.0430000	444
Future Price	1690.75	842.6669640	11.0000000	3757.00	444

Figure 1: Descriptive Statistics

Since these are provided by different stations, adaptations are made to extract only one information for each month of the year: this justifies both the presence of the variables T2MMAX and T2MMIN and the use of the mean as position index for all the variables.

Moreover, monthly US Cocoa historical Future Prices are collected and descriptive statistics computed with SAS are provided for each variable in Figure 1.

The following graphs have been generated during data discovery:

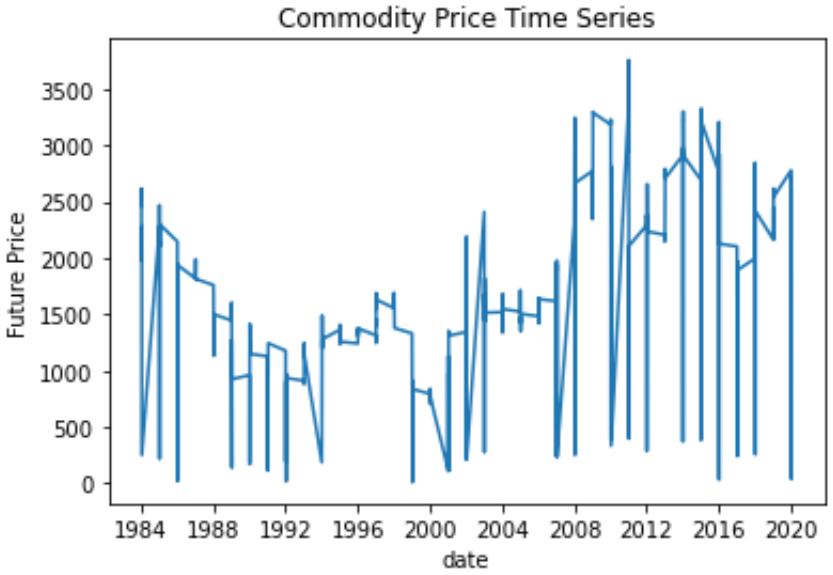

Figure 2: Time series of Cocoa's future prices

m

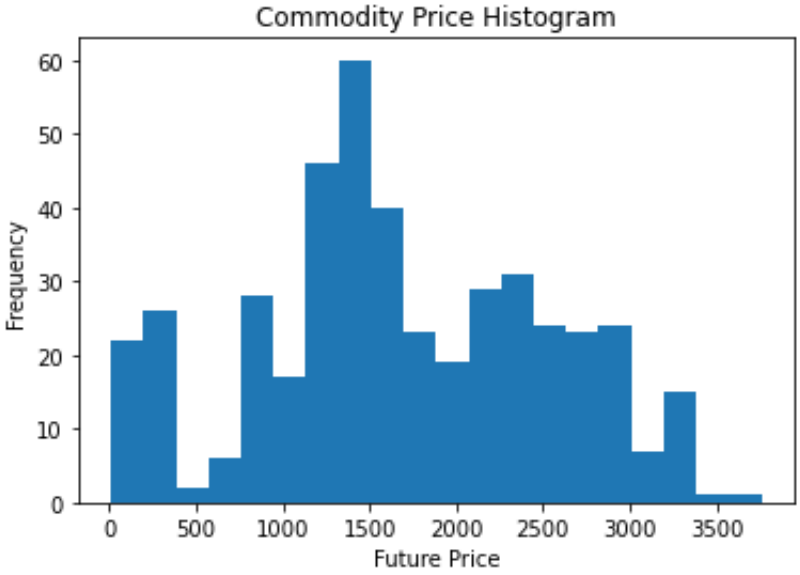

Figure 3: Histogram of Cocoa's future prices

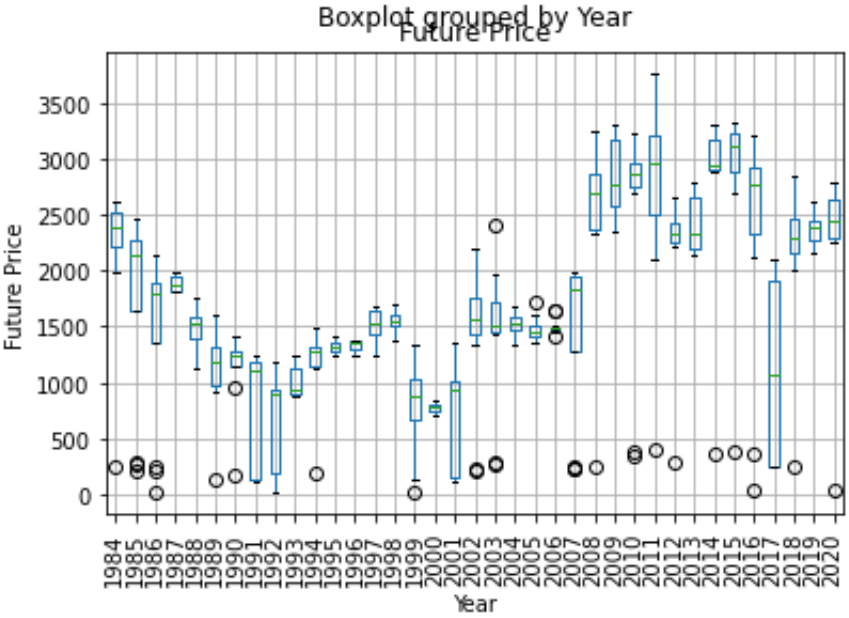

Figure 4: Boxplot grouped per year of Cocoa's future prices

3 The models

3.1 ARIMAX model

The use of statistical models to predict and understand the movement of cocoa prices is important for market participants and policymakers. In this paper, we propose the use of ARIMAX (AutoRegressive Integrated Moving Average with eXogenous variables) to do price discovery on cocoa based on T2M, PRECTOTCORR, and QV2M variables.

ARIMAX is a powerful time-series model that allows for the inclusion of exogenous variables that can help improve the accuracy of the forecasts. ARIMAX models are widely used in economics, finance, and other fields to model and forecast time-series data. ARIMAX combines the features of ARIMA (AutoRegressive Integrated Moving Average) and regression models.

The inclusion of these exogenous variables in the ARIMAX model can help capture the effect of external factors on cocoa prices and improve the accuracy of the forecasts. The use of ARIMAX can provide market participants and policymakers with valuable insights into the factors that influence cocoa prices and help them make informed decisions.

We excluded the maximum and minimum temperatures to avoid collinearity problem, having already on our model the mean temperature as a proxy for temperature-related information. Performing such analysis, using Python Programming, we obtained the following results:

Table 1: SARIMAX Results

Dep. Variable:	Future Price$_{C}$*ocoa*	No. Observations:	444	
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 12)	Log Likelihood	-3385.709	
Date:	Fri, 28 Apr 2023	AIC	6785.418	
Time:	17:30:19	BIC	6813.881	
Sample:	0 - 444	HQIC	6796.656	
Covariance Type:	opg			
	coef	std err	z	P¿—z—
PRECTOTCORR	24.0710	30.821	0.781	0.435
OV2M	-62.7587	38.811	-1.617	0.106
T2M	155.5292	59.794	2.601	0.009
ar.L1	0.1459	0.045	3.219	0.001
ma.L1	-0.8379	0.027	-31.284	0.000
ma.S.L12	-0.9139	0.018	-51.963	0.000
sigma2	3.753e+05	1.47e+04	25.519	0.000
Ljung-Box (L1) (Q): 0.02				
Prob(Q): 0.89				
Heteroskedasticity (H): 3.05				
Prob(H) (two-sided): 0.00				
Jarque-Bera (JB): 608.88				
Prob(JB): 0.00				
Kurtosis: 7.82				

The table represents the summary output of an ARIMAX model with dependent variable y and three independent variables. The model used an order of (1,1,1) for the ARIMA part and (0,1,1,12) for the seasonal part. The number of observations used in the model is 444.

The coefficients section of the table shows the estimated values of the regression coefficients for each independent variable and for the ARIMA and seasonal components. The p-values for each coefficient determine the significance of the variable. A p-value less than the chosen significance level indicates a significant relationship between the independent variable and the dependent variable. In this case, the coefficient for T2M has a p-value of 0.009, indicating that it is significant in the model.

The Ljung-Box statistic measures the independence of the residuals, and a value less than the significance level indicates a lack of fit in the model. In this case, the Ljung-Box test has a p-value of 0.89, indicating that the residuals are independent.

The Jarque-Bera test measures the normality of the residuals, and a p-value less than the significance level indicates a lack of normality. In this case, the JB test has a p-value of 0.00, indicating that the residuals are not normally distributed.

The Heteroskedasticity test checks for the homogeneity of variance in the residuals. The p-value of this test is also less than the significance level, indicating that there is heteroskedasticity in the residuals.

Overall, the model has a significant relationship between the dependent and independent variables, but there is evidence of a lack of fit, non-normality of residuals, and heteroskedasticity. Therefore, it is important to further investigate the model and potentially consider alternative modeling approaches.

3.2 LSTM Model

In attempting to find valuable connections between our natural variables and the future price of cocoa, we decided to try and use a machine learning tool, the LSTM Model.
The Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN) that is commonly used in the field of deep learning for time series analysis, natural language processing, and speech recognition.

LSTM models are designed to capture long-term dependencies and patterns in sequential data, such as time series. Unlike traditional feedforward neural networks, LSTM networks contain feedback connections that enable them to retain information over long periods of time.

At each time step, the LSTM model receives an input vector and outputs a hidden state vector. The input vector is combined with the previous hidden state and fed through a series of gates that control the flow of information in and out of the cell. The gates are comprised of sigmoid activation functions and decide which information to keep or forget, as well as which information to output.

The LSTM model contains three types of gates:

- Forget gate: controls which information from the previous hidden state should be forgotten
- Input gate: controls which new information should be added to the cell
- Output gate: controls which information from the cell should be output to the next time step

By learning which information to keep or forget at each time step, the LSTM model is able to capture long-term dependencies and patterns in the data.

LSTM models have been shown to be effective in a wide range of applications, including natural language processing, speech recognition, image captioning, and time series analysis. They are particularly useful for tasks that involve

sequential data and where the relationships between the data points are complex and non-linear.
Using the same natural variables (T2M, PRECTOTCORR, QV2M,T2MAX,T2MIN) to perform an LSTM model on cocoa future prices could be a good approach for several reasons:

- Non-linearity: LSTMs are particularly effective at capturing non-linear relationships between variables, which is important in financial time series analysis where the relationships between variables can be complex and non-linear.
- Time dependence: LSTMs can handle time dependence in the data, which is a critical aspect of financial time series analysis. This allows the model to learn from past observations and make predictions based on the historical patterns.
- Feature selection: LSTMs can effectively select and weigh important features in the data. This is particularly useful in financial analysis where a large number of variables may be available, and it can be challenging to identify the most relevant features.
- Robustness: LSTMs are robust to missing data and noisy observations, which is important in financial time series analysis where data may be incomplete or contain errors.

Overall, using the same variables in an LSTM model to predict cocoa future prices could be a good approach due to its ability to capture non-linear relationships, handle time dependence, perform feature selection, and be robust to missing data and noisy observations.

We split the dataset in a 80:20 ratio. The first 80% of the data will be used for training the LSTM model and the remaining 20% for testing and validating the trained model. Reshaping is carried out because the LSTM model requires input data in 3D format.

In performing the fit of the model to the data we have made the following parameterization decision:

- Optimizer = Adam
  Adam is a commonly used optimizer that works well for a wide range of problems. It is an adaptive learning rate optimization algorithm that is efficient and converges quickly
- Batch size = 156
  Used a larger batch size to potentially achieve faster convergence
- Number of epochs = 300 The number of epochs depends on how quickly the model is converging and how well it is generalizing to the validation set.

A graph of the training process is displayed below:

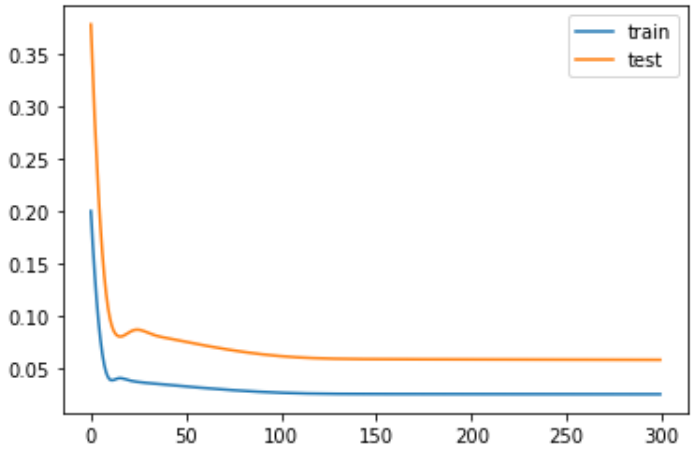

Figure 5: Plot of training errors over epochs

In order to evaluate the fit of our model to the data, we then perform the following analysis:

- RMSE The RMSE metric is calculated by taking the square root of the average of the squared differences between the predicted and actual values. Therefore, a lower RMSE value indicates better performance of the model in predicting the values.
- Trends and patterns: LSTM models are especially useful for capturing complex temporal patterns and trends in the data. Therefore, it's important to examine the predictions and see if they match up with any known trends or patterns in the data.

As per the RMSE metric, we obtain 0.23 in the standardized dataset, where the value range from 0 to 1, and 874.05 in the rescaled dataset, figuring the real scale of cocoa's future price data.

In this case, a test RMSE value of 903.138 indicates that, on average, the LSTM model's predictions for cocoa prices have an error of approximately 874.05. In other words, the model's predictions are off by an average of 874.05 compared to the actual prices of cocoa.
It's important to note that the interpretation of the RMSE value depends on the scale and context of the data being predicted. In the context of cocoa prices, an RMSE value of 874.05 may be considered high or low depending on factors such as the volatility of cocoa prices and the level of accuracy required for the specific application.

It can be helpful to look at the accuracy of individual predictions and see how well they match up with the actual values. This can be done by plotting the predicted values against the actual values, or by looking at specific examples where the model made predictions.

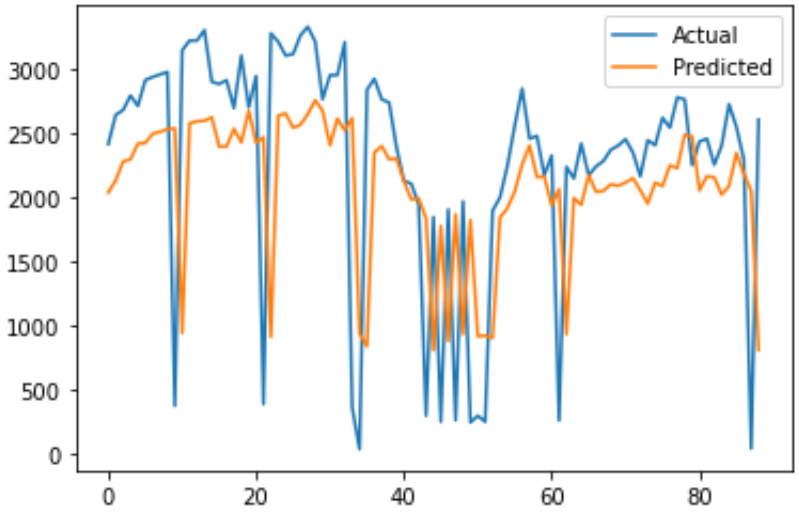

Figure 6: Actual vs predicted for Cocoa's future prices

While the model has a clear lack of ability in fitting the data correctly and has not had enough exact predicting power, it seems able to capture correctly the general trends and patterns.

It is important to note that this analysis has been performed with the sole scope of learning, being very rudimental and lacking important pre-fitting analysis that could help us obtain a better fit.

More data could have been inserted in the model and we could have done a better job in tuning the hyperparameters of out model to your specific dataset and problem. This involves trying different combinations of hyperparameters and evaluating their performance on a validation set.

4 APPENDIX 1 - PYTHON CODE

```
######### AUTHOR OF THE CODE: VITTORIO BALESTRIERI #################


# load required libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from numpy import concatenate
from pandas import DataFrame
from pandas import concat
from matplotlib import pyplot
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM



# Load data
commodity_data = pd.read_csv("data_cocoa_;.csv",delimiter=";")
commodity_data["date"] = pd.to_datetime(commodity_data["date"])

commodity_data["Year"] = pd.DatetimeIndex(commodity_data["date"]).year

cocoa_data = commodity_data.iloc[:,2:8]
cols = cocoa_data.columns.tolist()
cols = cols[-1:] + cols[:-1]
cocoa_data = cocoa_data[cols]
# Create box plot of prices by year
commodity_data.boxplot(column="Future Price", by="Year")
plt.xlabel("Year")
plt.ylabel("Future Price")
plt.xticks(rotation=90)
plt.show()

# Plot histogram of prices
plt.hist(commodity_data["Future Price"], bins=20)
plt.xlabel("Future Price")
plt.ylabel("Frequency")
plt.title("Commodity Price Histogram")
plt.show()
```

```
# Plot time series of prices
plt.plot(commodity_data["date"], commodity_data["Future Price"])
plt.xlabel("date")
plt.ylabel("Future Price")
plt.title("Commodity Price Time Series")
plt.show()

# Define features and target variable
features = commodity_data.drop(columns=["Future Price","Year","Month","date"]).values
target = commodity_data["Future Price"].values

test1 = features[:,(0,1,2)]


# Define ARIMA model
order = (1, 1, 1)
seasonal_order = (0, 1, 1, 12)


# Define model
model = sm.tsa.statespace.SARIMAX(endog=target, exog=test1, order=order,
seasonal_order=seasonal_order)

# Fit model
results = model.fit()

# Print results
print(results.summary())


################################################
#### LSTM MODEL ############

num_features = 6 #Number of features in the dataset
lag_steps = 1 #Number of lagged time features to be generated
label_feature = 'Future Price' #The column in dataset that model is being built to predict

# This function arranges the dataset to be used for surpervised
learning by shifting the input values of features by the number

def sequential_to_supervised(data, lag_steps = 1, n_out = 1, dropnan = True):
    features = 1 if type(data) is list else data.shape[1] # Get the number of features in da
    df = DataFrame(data)
    cols = list()
    feature_names = list()
```

```
    for i in range(lag_steps, 0, -1):
        cols.append(df.shift(i)) # This will be the shifted dataset
        feature_names += [(str(df.columns[j])) + '(t-%d)' % (i) for j in range(features)] #
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    if i == 0:
        feature_names += [(str(df.columns[j])) + '(t)' for j in range(features)] # Names of
    else:
        feature_names += [(str(df.columns[j])) + '(t+%d)' % (i) for j in range(features)] #
    agg = concat(cols, axis=1)
    agg.columns = feature_names
    if dropnan:
        agg.dropna(inplace=True)
    return agg

supervised_dataset = sequential_to_supervised(cocoa_data, lag_steps)

# Move label column to the end of dataset
cols_at_end = [label_feature + '(t)']
supervised_dataset = supervised_dataset[[c for c in
supervised_dataset if c not in cols_at_end] + [c for c in
cols_at_end if c in supervised_dataset]]

# Dropping the current timestep columns of features other than the one being predicted
supervised_dataset.drop(supervised_dataset.columns[(num_features*lag_steps) :
(num_features*lag_steps + num_features -1)], axis=1, inplace=True)
scaler = MinMaxScaler(feature_range=(0, 1))
supervised_dataset_scaled = scaler.fit_transform(supervised_dataset) # Scaling all values

split = int(supervised_dataset_scaled.shape[0]*0.8) # Splitting for traning and testing
train = supervised_dataset_scaled[:split, :]
test = supervised_dataset_scaled[split:, :]

train_X, train_y = train[:, :-1], train[:, -1] # The label column is separated out
test_X, test_y = test[:, :-1], test[:, -1]

train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1])) # Reshaping done
for LSTM as it needs 3D input
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))

# Defining the LSTM model to be fit
model = Sequential()
model.add(LSTM(85, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
# Fitting the model
history = model.fit(train_X, train_y, epochs=300, batch_size=156,
validation_data=(test_X, test_y), verbose=2, shuffle=False)
# Plotting the training progression
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()


# Using the trained model to predict the label values in test dataset
yhat = model.predict(test_X)

# Reshaping back into 2D
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))

# Concatenating the predict label column with Test data input
features
inv_yhat = concatenate((test_X[:, 0:], yhat), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat) # Rescaling back

inv_yhat = inv_yhat[:, num_features*lag_steps] # Extracting the rescaled predicted
label column

test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_X[:, 0:], test_y), axis=1) # Re joing the
test dataset for inversing the scaling
inv_y = scaler.inverse_transform(inv_y) # Rescaling the actual label column values
inv_y = inv_y[:, num_features*lag_steps] # Extracting the rescaled actual label column

# Calculating RMSE
rmse = np.sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
rmse2 = np.sqrt(mean_squared_error(test_y, yhat))

# Plotting Actual vs Predicted
pyplot.plot(inv_y, label = 'Actual')
pyplot.plot(inv_yhat, label = 'Predicted')
pyplot.legend()
pyplot.show()
```