**Algorithms and their details**

**Linear Regression Algorithm**

In CryptOL™ linear regression is used to extrapolate a trend from Yahoo Finance library assets. Linear regression and ordinary least squares (OLS) are decades-old statistical techniques that can be incorporated into machine learning to extrapolate a trend in the observed asset and predict the direction of future price movement.

Anvil Code:

result = alert(content="In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). \n\n In CryptOL™ linear regression is used to extrapolate a trend from Yahoo Finance library asset. Linear regression and ordinary least squares (OLS) are decades-old statistical techniques that can be incorporated into machine learning to extrapolate a trend in the observed asset and predict the direction of future price movement.",

title="Linear Regression Algorithm",

large=True,

buttons=[

("Close", "CLOSE")

])

**Server side Linear Regression code:**

import anvil.stripe

import predict

import anvil.server

from sklearn import linear\_model

@anvil.server.callable

def getLatest():

latest = predict.downloadLatest()

current = predict.recentPrice(latest)

f\_current = "$" + str("{:,.2f}".format(current))

return f\_current

@anvil.server.callable

def getPrediction(interval = "15 min"):

reg = linear\_model.LinearRegression()

if(interval == "1 hour"):

historical = predict.downloadHistorical("7d","1h")

print(interval)

if(interval == "1 day"):

historical = predict.downloadHistorical("90d","1d")

print(interval)

else:

historical = predict.downloadHistorical()

print(interval)

cleaned = predict.cleanHistorical(historical)

X = predict.getX(cleaned)

y = predict.getY(cleaned)

X\_train, X\_test, y\_train, y\_test = predict.split(X,y)

reg.fit(X\_train,y\_train)

current = predict.downloadLatest()

prediction = reg.predict(current)

f\_prediction = "$" + str("{:,.2f}".format(prediction[-1]))

current['prediction'] = prediction

accuracy = 0

for x in range(1,len(current) - 1):

if ((current['prediction'].iloc[x] > current.iloc[x][('Close','BTC-USD')] and (current.iloc[x + 1][('Close','BTC-USD')] > current.iloc[x][('Close','BTC-USD')]))):

accuracy += 1

elif ((current['prediction'].iloc[x] < current.iloc[x][('Close','BTC-USD')] and (current.iloc[x + 1][('Close','BTC-USD')] < current.iloc[x][('Close','BTC-USD')]))):

accuracy += 1

print(len(current))

print("accuracy - ", accuracy/len(current) \* 100)

return (accuracy/len(current) \* 100), prediction, f\_prediction

**Client side Linear Regression code:**

def refresh\_linear\_click(self, \*\*event\_args):

"""This method is called when the button is clicked"""

accuracy, prediction, formatted = anvil.server.call('getPrediction',self.drop\_down\_linear.selected\_value)

self.predicted\_linear.text = formatted

self.accuracy\_linear.text = str("{:,.2f}".format(accuracy))+"%"

#Below lines populate the Suggestion field, if the accuracy is over 50% and predicted price is over the current price we get a buy result

if ((formatted>anvil.server.call('getLatest')) and (accuracy>50)):

self.suggestion\_linear.foreground = "#85bb65"

self.suggestion\_linear.text = "BUY"

else:

self.suggestion\_linear.foreground = "#FF6347"

self.suggestion\_linear.text = "HOLD"

**Time Series**

A famous and widely used forecasting method for time-series prediction is the Auto-Regressive Integrated Moving Average (ARIMA) model. CryptOL™ ARIMA model is capable of capturing a suite of different standard temporal structures in time-series data.

Anvil Code:

result = alert(content=" Time series forecasting is used to predict future values based on previously observed values and one of the best tools for trend analysis and future prediction. \n\n A famous and widely used forecasting method for time-series prediction is the Auto-Regressive Integrated Moving Average (ARIMA) model. CryptOL™ ARIMA model is capable of capturing a suite of different standard temporal structures in time-series data.",

title="**Time Series**",

large=True,

buttons=[

("Close", "CLOSE")

])

**Server side Time Series code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pandas.plotting import lag\_plot

from pandas import datetime

from statsmodels.tsa.arima\_model import ARIMA

from sklearn.metrics import mean\_squared\_error

import predict

import anvil.server

import locale

locale.setlocale(locale.LC\_ALL, '')

import anvil.mpl\_util

@anvil.server.callable

def arima(interval):

if (interval == "1 hour"):

df = predict.ts\_download\_btc("4d", "1h")

if (interval == "1 day"):

df = predict.ts\_download\_btc("90d", "1d")

else:

df = predict.ts\_download\_btc("2d", "15m")

train\_data, test\_data = df[0:int(len(df) \* 0.7)], df[int(len(df) \* 0.7):]

training\_data = train\_data.values

test\_data = test\_data.values

history = [x for x in training\_data]

model\_predictions = []

N\_test\_observations = len(test\_data)

for time\_point in range(N\_test\_observations):

model = ARIMA(history, order=(4, 1, 0))

model\_fit = model.fit(disp=0)

output = model\_fit.forecast()

yhat = output[0]

model\_predictions.append(yhat)

true\_test\_value = test\_data[time\_point]

history.append(true\_test\_value)

pred = []

for x in model\_predictions:

pred.append(x[0])

accuracy = 0

for x in range(len(test\_data) -1 ):

if (pred[x] > test\_data[x] and test\_data[x + 1] > test\_data[x]):

accuracy += 1

elif (pred[x] < test\_data[x] and test\_data[x + 1] < test\_data[x]):

accuracy += 1

print("size = ", len(test\_data))

print("accuracy = ", accuracy)

return (accuracy/len(test\_data)\*100),test\_data, pred

@anvil.server.callable

def ts\_plot(x, y):

plt.figure(1, figsize=(10,5))

plt.plot(x, 'crimson')

plt.plot(y, 'blue')

# Return this plot as a PNG image in a Media object

return anvil.mpl\_util.plot\_image()

**Client Side Time Series Code:**

def refresh\_arima\_click(self, \*\*event\_args):

accuracy, history, predictions = anvil.server.call('arima',self.drop\_down\_arima.selected\_value)

self.predicted\_arima.text = "$" + str("{:,.2f}".format(predictions[-1]))

self.accuracy\_arima.text = str("{:,.2f}".format(accuracy))+"%"

suggested= str("{:.2f}".format(predictions[-1]))

accura = int(accuracy)

scall = anvil.server.call('getLatest')

#Below lines populate the Suggestion field, if the accuracy is over 50% and predicted price is over the current price we get a buy result

if ((suggested>scall) and (accura>50)):

self.suggestion\_arima.foreground = "#85bb65"

self.suggestion\_arima.text = "BUY"

else:

self.suggestion\_arima.foreground = "#FF6347"

self.suggestion\_arima.text = "HOLD"

**Long Short-Term Memory (LSTM)**

In CryptOL™ LSTM is very powerful in sequence prediction problems because it is able to store past information. This is important in our case because the previous price of BitCoin is crucial in predicting its future price.

Anvil Code:

result = alert(content="LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. \n\n In CryptOL™ LSTM is very powerful in sequence prediction problems because it is able to store past information. This is important in our case because the previous price of BitCoin is crucial in predicting its future price.",

title="**Long Short-Term Memory (LSTM)**",

large=True,

buttons=[

("Close", "CLOSE")

])

**Server Side LSTM Code:**

import yfinance as yf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import predict

import anvil.server

import locale

locale.setlocale(locale.LC\_ALL, '')

import anvil.mpl\_util

tick = ["BTC-USD"]

def ts\_download\_btc(per="3d", inter="15m"):

data = yf.download(

tickers = tick,

period = per,

interval = inter,

auto\_adjust = True,

prepost = True,

threads = True,

proxy = None

)

return data[["Open","High","Close"]]

@anvil.server.callable

def LSTMpred():

df = ts\_download\_btc("3d")

df.fillna(method="ffill", inplace=True)

df.dropna(inplace=True)

print(df)

cl = df

train = cl[0:int(len(cl)\*0.80)]

scl = MinMaxScaler()#scale the data

scl.fit(train.values.reshape(-1,1))

cl =scl.transform(cl.values.reshape(-1,1))

def processData(data,lb):

X,Y = [],[]

for i in range(len(data)-lb-1):

X.append(data[i:(i+lb),0])

Y.append(data[(i+lb),0])

return np.array(X),np.array(Y)

lb=10

X,y = processData(cl,lb)

X\_train,X\_test = X[:int(X.shape[0]\*0.90)],X[int(X.shape[0]\*0.90):]

y\_train,y\_test = y[:int(y.shape[0]\*0.90)],y[int(y.shape[0]\*0.90):]

model = Sequential()

model.add(LSTM(256,input\_shape=(lb,1)))

model.add(Dense(1))

model.compile(optimizer='adam',loss='mse')

#reshaping data

X\_train = X\_train.reshape((X\_train.shape[0],X\_train.shape[1],1))

X\_test = X\_test.reshape((X\_test.shape[0],X\_test.shape[1],1))

history = model.fit(X\_train,y\_train,epochs=80,validation\_data=(X\_test,y\_test),shuffle=False)

model.summary()

Xt = model.predict(X\_train)

#train data

Xt = model.predict(X\_test)

#test data

pred = scl.inverse\_transform(Xt).tolist()

acc = 0

for x in range(len(Xt)-1):

if(pred[x] > Xt[x] and Xt[x+1]>Xt[x]):

acc += 1

elif (pred[x] < Xt[x] and Xt[x+1]<Xt[x]):

acc += 1

accu = (acc/len(Xt)\*100)

#print(pred[-1])

#print(accu)

prediction = pred[-1]

return accu, prediction

#print("prediction: ")

#print("accu = ", accu)

**Client Side LSTM Code:**

def refresh\_lstm\_click(self, \*\*event\_args):

accu, prediction = anvil.server.call('LSTMpred')

self.accuracy\_lstm.text = str("{:.2f}".format(accu))+"%"

stripper = str(prediction)

newstrp=stripper.strip("[]")

self.predicted\_lstm.text = "$" + newstrp

#Below lines populate the Suggestion field, if the accuracy is over 50% and predicted price is over the current price we get a buy result

if ((newstrp>anvil.server.call('getLatest')) and (accu>50)):

self.suggestion\_lstm.foreground = "#85bb65"

self.suggestion\_lstm.text = "BUY"

else:

self.suggestion\_lstm.foreground = "#FF6347"

self.suggestion\_lstm.text = "HOLD"

**Logistic Regression**

In CryptOL™ Logistic Regression algorithm is a part of the Supervised Learning method of Machine Learning. It is a statistical method for the analysis of a dataset. It has one or more independent variables that determine an outcome. We use a training set and a test set of data to predict the selling price of BitCoin.

Anvil Code:

result = alert(content="Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. \n\n In CryptOL™ Logistic Regression algorithm is a part of the Supervised Learning method of Machine Learning. It is a statistical method for the analysis of a dataset. It has one or more independent variables that determine an outcome. We use a training set and a test set of data to predict the selling price of BitCoin.",

title="**Logistic Regression Algorithm**",

large=True,

buttons=[

("Close", "CLOSE")

])

def button\_10\_click(self, \*\*event\_args):

alert("This algorithm is currently in development. Please check back in 2021 Q3.")

pass

**Survival Modeling**

CryptOL™ adopts Cox's hazard model to predict BitCoin’s future rising or dropping probabilities. Specifically, we define the problem of Buy-and-Sell-Point Prediction from the survival analysis perspective. We apply the trained model for the cryptocurrency market forecasting on two cryptocurrencies traded on the CoinBase exchange.

Anvil Code:

result = alert(content="Survival analysis is a statistical method that aims to predict the time to an event, such as death, the diagnosis of a disease or the failure of a mechanical part. \n\n CryptOL™ adopts Cox's hazard model to predict BitCoin’s future rising or dropping probabilities. Specifically, we define the problem of Buy-and-Sell-Point Prediction from the survival analysis perspective. We apply the trained model for the cryptocurrency market forecasting on two cryptocurrencies traded on the CoinBase exchange.",

title="**Survival Modeling Algorithm**",

large=True,

buttons=[

("Close", "CLOSE")

])

def button\_10\_click(self, \*\*event\_args):

alert("This algorithm is currently in development. Please check back in 2021 Q3.")

pass

**CryptOL™ Proprietary Algorithm**

Our “secret” proprietary algorithm incorporates the best in prediction techniques of all the standardized algorithms combined to produce the most accurate comprehensive value that is possible through Machine Learning.

Anvil Code:

result = alert(content="Our “secret” proprietary algorithm incorporates the best in prediction techniques of all the standardized algorithms combined to produce the most accurate comprehensive value that is possible through Machine Learning.",

title="**CryptOL™ Proprietary Algorithm**",

large=True,

buttons=[

("Close", "CLOSE")

])

def button\_10\_click(self, \*\*event\_args):

alert("This algorithm is currently in development. Please check back in 2021 Q3.")

pass