FIN42110: HomeWork 6

Ross Kearney

14th April 2022

1 underlying asset

The underlying asset chosen was Apple (AAPL)

2 predictor variables

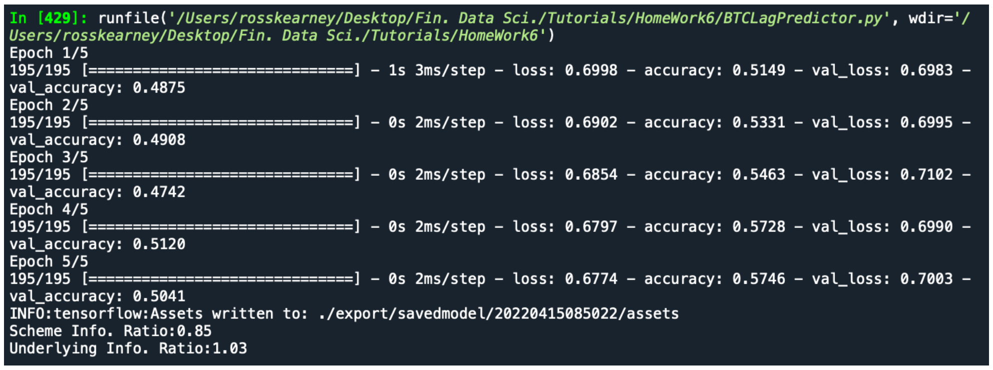
The predictor variable added was a moving average indicator. The 5 and 10 day moving averages were found, and the binary signal for the indicator was triggered when the 5 and 10 day moving averages crossed.

3 Model improvement

To improve the model, a few minor adjustments were made.

Firstly, a moving average indicator was created using 5 and 10 day rolling averages.
Secondly, the test size was adjusted to 30% of the overall data set, allowing more training to occur.
Third, the existing lag indicator was changed to a 2 day lag.

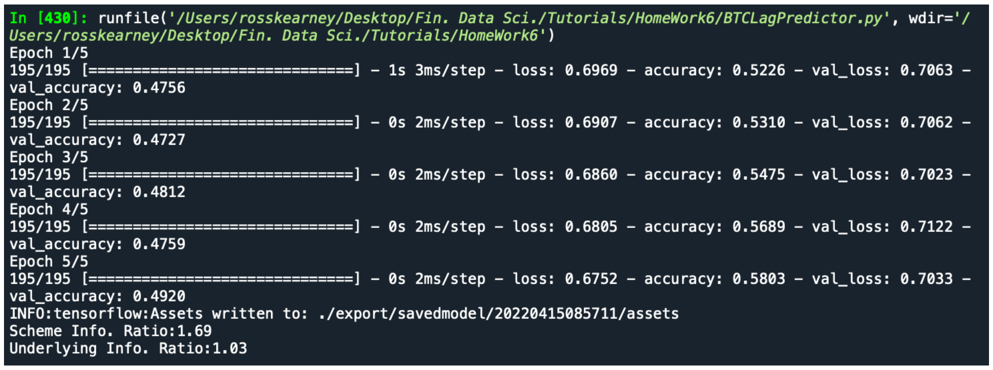
4 Results



```
In [429]: runfile('/Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6/BTCLagPredictor.py', wdir='/
Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6')
Epoch 1/5
195/195 [==============================] - 1s 3ms/step - loss: 0.6998 - accuracy: 0.5149 - val_loss: 0.6983 -
val_accuracy: 0.4875
Epoch 2/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6902 - accuracy: 0.5331 - val_loss: 0.6995 -
val_accuracy: 0.4908
Epoch 3/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6854 - accuracy: 0.5463 - val_loss: 0.7102 -
val_accuracy: 0.4742
Epoch 4/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6797 - accuracy: 0.5728 - val_loss: 0.6990 -
val_accuracy: 0.5120
Epoch 5/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6774 - accuracy: 0.5746 - val_loss: 0.7003 -
val_accuracy: 0.5041
INFO:tensorflow:Assets written to: ./export/savedmodel/20220415085022/assets
Scheme Info. Ratio:0.85
Underlying Info. Ratio:1.03
```

Figure 1: Predictor model result for Apple (AAPL) with no model adjustment

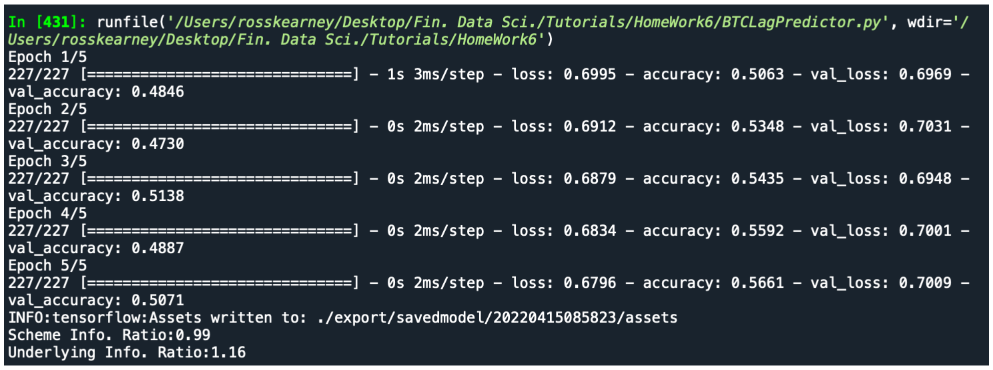
Here is the results of the original model used to predict the movement of AAPL. It shows modest predictive accuracy and a good Underlying Info. Ratio.



```
In [430]: runfile('/Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6/BTCLagPredictor.py', wdir='/
Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6')
Epoch 1/5
195/195 [==============================] - 1s 3ms/step - loss: 0.6969 - accuracy: 0.5226 - val_loss: 0.7063 -
val_accuracy: 0.4756
Epoch 2/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6907 - accuracy: 0.5310 - val_loss: 0.7062 -
val_accuracy: 0.4727
Epoch 3/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6860 - accuracy: 0.5475 - val_loss: 0.7023 -
val_accuracy: 0.4812
Epoch 4/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6805 - accuracy: 0.5689 - val_loss: 0.7122 -
val_accuracy: 0.4759
Epoch 5/5
195/195 [==============================] - 0s 2ms/step - loss: 0.6752 - accuracy: 0.5803 - val_loss: 0.7033 -
val_accuracy: 0.4920
INFO:tensorflow:Assets written to: ./export/savedmodel/20220415085711/assets
Scheme Info. Ratio:1.69
Underlying Info. Ratio:1.03
```

Figure 2: Predictor model results for Apple with moving average indicator

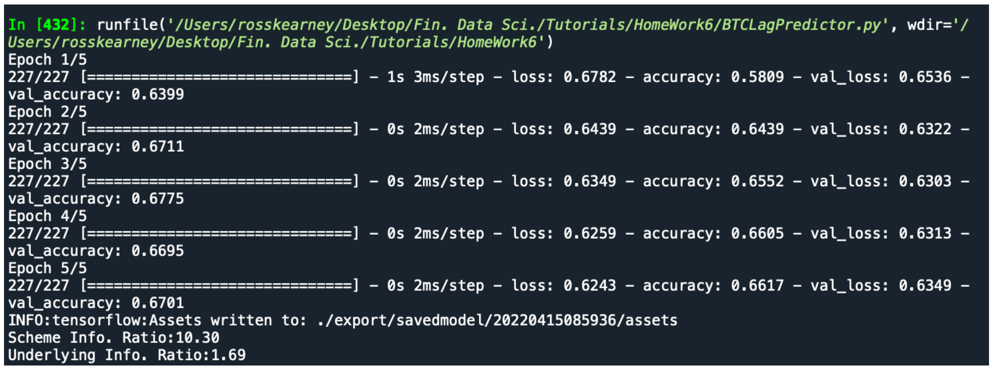
These results show the inclusion of the 5 and 10 day moving average indicator, and show little improvement in the predictive accuracy but a large improvement in the Scheme Info. Ratio.



```
In [431]: runfile('/Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6/BTCLagPredictor.py', wdir='/
Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6')
Epoch 1/5
227/227 [==============================] - 1s 3ms/step - loss: 0.6995 - accuracy: 0.5063 - val_loss: 0.6969 -
val_accuracy: 0.4846
Epoch 2/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6912 - accuracy: 0.5348 - val_loss: 0.7031 -
val_accuracy: 0.4730
Epoch 3/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6879 - accuracy: 0.5435 - val_loss: 0.6948 -
val_accuracy: 0.5138
Epoch 4/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6834 - accuracy: 0.5592 - val_loss: 0.7001 -
val_accuracy: 0.4887
Epoch 5/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6796 - accuracy: 0.5661 - val_loss: 0.7009 -
val_accuracy: 0.5071
INFO:tensorflow:Assets written to: ./export/savedmodel/20220415085823/assets
Scheme Info. Ratio:0.99
Underlying Info. Ratio:1.16
```

Figure 3: Predictor model results for Apple with 30% test size

The next model incorporated an adjustment to the test size. The training sample was increased from 60% of the data set to 70%. This caused minor improvements in the predictive accuracy, but a considerable improvement in the Underlying Info. Ratio.



```
In [432]: runfile('/Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6/BTCLagPredictor.py', wdir='/
Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork6')
Epoch 1/5
227/227 [==============================] - 1s 3ms/step - loss: 0.6782 - accuracy: 0.5809 - val_loss: 0.6536 -
val_accuracy: 0.6399
Epoch 2/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6439 - accuracy: 0.6439 - val_loss: 0.6322 -
val_accuracy: 0.6711
Epoch 3/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6349 - accuracy: 0.6552 - val_loss: 0.6303 -
val_accuracy: 0.6775
Epoch 4/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6259 - accuracy: 0.6605 - val_loss: 0.6313 -
val_accuracy: 0.6695
Epoch 5/5
227/227 [==============================] - 0s 2ms/step - loss: 0.6243 - accuracy: 0.6617 - val_loss: 0.6349 -
val_accuracy: 0.6701
INFO:tensorflow:Assets written to: ./export/savedmodel/20220415085936/assets
Scheme Info. Ratio:10.30
Underlying Info. Ratio:1.69
```

Figure 4: Predictor model results for Apple lag increased to 2 days

The third and final adjustment was to increase the lag indicator from one day to two days. This had the biggest effect on the predictive ability of the model. It significantly increased the predictive accuracy of the model and also significantly increased the Underlying Info. Ratio.

5 Python Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  7 11:11:46 2022

@author: rosskearney
"""

import pandas as pd
import numpy as np
import datetime
import shutil
import os
import tensorflow as tf
import pandas_ta as ta
from sklearn.model_selection import train_test_split

def create_model():
    # Random seed for repeatability
  np.random.seed(100)
  tf.random.set_seed(100)
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu',input_dim = len(cols)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
  ])

df = pd.DataFrame() # Empty DataFrame
# df = df.ta.ticker("BTC-USD")
# df = df.ta.ticker("JPYUSD=X")

df = df.ta.ticker('AAPL')

df['Return'] = np.log(df['Close']/df['Close'].shift(2))

# create lagged returns
cols=[]
lags = range(1,50)
for lag in lags:
    df[f'retlag{lag}'] = df['Return'].shift(lag)
    col =f'retlagdir{lag}'
    df[col] = np.where(df['Return'].shift(lag)> 0, 1, 0)
    cols.append(col)

df.dropna(inplace=True)

df['direction'] = np.where(df['Return'] > 0, 1, 0)

# =============================================================================
#
# =============================================================================


# Create rolling averages, easy to change
df['CURrolling10'] = df['Close'].rolling(10).mean()
df['CURrolling5'] = df['Close'].rolling(5).mean()

# drop empty values
df.dropna(inplace = True)

# Create new column, if rolling 5 is > rolling 30, plot a 'buy' signal
# if rolling 5 < rolling 30, plot 'sell' signal
# (plot runs to high/low of highest/lowest stock price for clean graph)
```

```
df['CrossOver'] = np.where(df['CURrolling5'] > df['CURrolling10'], 1, 0)


# ============================================================================
#
# ============================================================================


# split the dataset in training and test datasets
train, test = train_test_split(df.dropna(), test_size=0.3, shuffle=False)


# create the model
model = create_model()

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
r = model.fit(train[cols],
              train['direction'],
              epochs=5,
              validation_data=(test[cols], test['direction']),
              callbacks=[tensorboard_callback]) #verbose=False
OUTPUT_DIR = "./export/savedmodel"
shutil.rmtree(OUTPUT_DIR,ignore_errors=True)
EXPORT_PATH = os.path.join(OUTPUT_DIR,datetime.datetime.now().strftime("%Y%m%d%H%M%S"))
tf.saved_model.save(model,EXPORT_PATH)

pred = model.predict(test[cols])
trade = pred > 0.5 + pred.std()
profits = test.loc[trade, 'Return']
print('Scheme Info. Ratio:' + "{:.2f}".format(np.sqrt(365)*profits.mean()/profits.std()))
print('Underlying Info. Ratio:' + "{:.2f}".format(np.sqrt(365)*test['Return'].mean()/test['Return'].std()))
```