Forecasting GBP/USD Price Action: A Machine Learning Approach

Abstract

This document aims to explore the effectiveness of different machine learning models in forecasting the price movement of the GBPUSD currency pairing within the foreign exchange (forex) market. Forex markets are characterised by their volatility and complexity, making it difficult to forecast price movements accurately. Machine learning models are able to parse large amounts of data, identify intricate patterns, and make predictions based on historical data; hence, they have become prevalent tools within the realm of financial forecasting. In this study, I will compare the performance of random forest, artificial neural network, and support vector machine models when it comes to forecasting price movement and whether they can outperform the buy-and-hold return through the use of a backtesting environment (Backtesting.py). The experiments conducted will look at how different parameters and indicators affect the respective models forecast accuracy. The results show that when sufficiently tuned, the models are able to outperform conventional buy-and-hold strategies for certain time horizons.

1- Introduction

1.1- Background

Foreign exchange is a global and decentralised market revolving around the trade of currencies. Among a plethora of different currency pairs, the GBP/USD (British pound against the US dollar) holds significant importance due to both the UK and US being major economies with significant trade and financial ties. The GBP/USD pairing directly affects various economic sectors, including international trade and investment flows. The pairing exhibits adverse price fluctuations based on various factors, including recent geopolitical events such as Brexit [1], economic indicators such as rising inflation rates, and monetary policy enforced by the Bank of England and the US Federal Reserve, just to name a few. The dynamic and volatile nature of this pairing has made it increasingly difficult for accurate price projections to be established; however, the advent of machine learning models has emerged as a potential way to increase the accuracy of forex price forecasts [2], including the GBP/USD pair. This document primarily focuses on using machine learning models to forecast GBP/USD price movements and using these forecasts in conjunction with a baseline forex strategy to see if they can potentially be of use to traders. Traditional forex trading methods employ statistical models that may have limitations when it comes to capturing the complex patterns and nonlinear relationships present in the currency markets. Machine learning models have the potential to model these limitations by leveraging their ability to parse large amounts of data, detect complex patterns, and adapt to constantly changing market conditions [3].

1.2- Aims

The primary objective of this project is to determine the efficacy of using machine learning models for forecasting the price movement of GBP/USD, specifically focusing on the development of classification models. Furthermore, the project aims to evaluate the performance of these models when integrated with a basic trading strategy by conducting rigorous backtesting using unseen data. By addressing these goals, the study aims to provide answers to the following key research questions:

1. Which classification model produces the highest accuracy in predicting price trends for GBP/USD?

2. Which model, when combined with the trading strategy, yields the greatest return on investment during backtesting?

3. Can ensemble classification models increase the accuracy of GBP/USD price forecasts compared to individual machine learning models?

2- Relevant literature

Over the years,various forecasting methods have been employed in the field of finance, with an increasing focus on machine learning techniques. In a survey paper published in 2016 [4], which explored the applicability of computational intelligence in financial markets,a multitude of methods specifically designed for predicting stock data were presented. Importantly, many of these methods can also be adapted and applied to the foreign exchange market, making them relevant for forecasting currency pairs such as GBP/USD.

2.1- ANN

After extensive research on some of these stock market models, artificial neural networks (ANN) were selected as one of the classification models that would be used to forecast price action. ANNs are able to capture complex non-linear relationships between data due to the fact that they contain layers of interconnected neurons that each have their own non-linear activation functions. ANNs are able to emulate complex relationships between input and output predictions by applying transformations to each of these neurons. In the literature, it was observed that ANNs outperformed statistical models that they were compared against when forecasting the market value of assets [5]. Neural networks can also effectively combat the vanishing gradient problem [6] when correctly tuned with the appropriate activation function, thus preventing slower or no learning at all through the model. It is, however, important to note some prevalent issues with this model and precautions that need to be taken throughout training. ANNs can be prone to overfitting due to forming complex relationships with the training data, thus failing to generalise well to unseen data. In addition to this, ANNS suffer from being sensitive to hyperparameters, where in order for optimal performance to be reached, extensive effort needs to be put in during the tuning process.

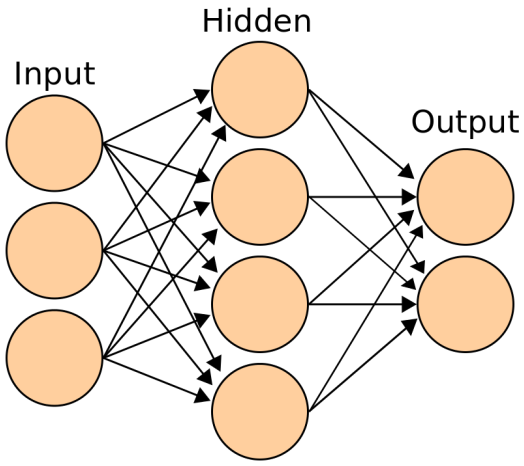

Figure 1: Basic ANN structure

2.2- Random Forest

Similar studies have been conducted on the performance of ensemble methods with regards to their ability to predict stock prices a day in advance [7]. They concluded that random forest regression and boosted regression trees performed the best, while support vector regression ensemble performed the worst in regards to mean absolute percentage error. Which is why I elected to use random forest (RF) as one of the classification models in my experiments over the other models mentioned in the study. This is due to its exceptional capability to make accurate predictions of time series data [8]. It uses a collection of decision tree predictors that each contain a unique subset of features in order to derive its output. For a classification problem, the most commonly occurring output among the individual decision tree predictors is chosen as the RF output.

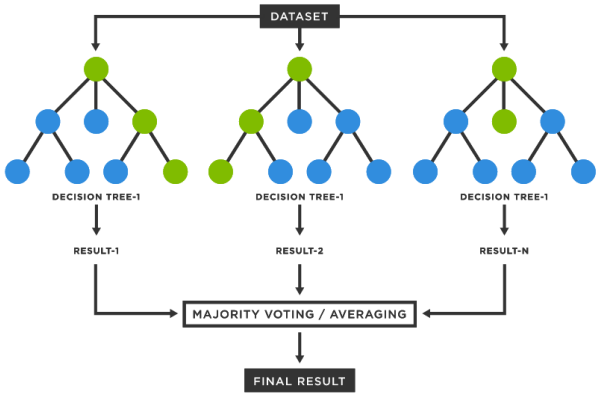

Figure 2: Random forest structure

2.3- SVM

The final classification model I elected to use was a support vector machine (SVM). This type of model is classified by its capacity to control the decision boundary for classifying entries and its use of kernel functions to transform input data into a computable form [9]. SVMs have been demonstrated to be resistant to overfitting, eventually producing high generalisation performance with regards to unseen data [9]. This model also solves the equivalent of linearly constrained quadratic problems during its training, such that the solution is always unique and contains a global optimum, this is in stark contrast to ANNs, which require non-linear activation functions that may get stuck in local optimums [9].

2.4- Environment

After looking into VectorBt, Backtrader, and Backtesting.py as all potential candidates for backtesting environments, I settled on using Backtesting.py. Through running my own tests, I found that the compilation and lookback speeds of running multiple iterations of the same backtest were fastest on Backtesting.py, followed by VectorBt and Backtrader. In addition to this, I found the documentation of Backtesting.py the easiest to follow when compared to the other two, where, in the case of Backtrader, it was no longer being supported. With VectorBt being a relatively new library, the documentation was fairly scarce, thus leading me to opt to use Backtesting.py as the environment to run my trading experiments.

3- Methodology

3.1- Data Collection

Price history data of the GBP/USD pairing was retrieved from Yahoo Finance[10] using the pandas-ta library[12] and converted into a dataframe to be ready for use. The dataframe initially contained ‘Date’, ‘Open’, ‘High’,’Low’, ‘Close’, ‘Volume’, ‘Dividends’ and ‘Stock Splits’ columns upon retrieval. The data ranged from 2003-12-01 to 2023-05-05

3.2- Pre-Processing

First, the ‘Volume’, ‘Dividends," and ‘Stock Splits’ columns were all deleted from the dataframe as they were not relevant to any of the indicators that would later be calculated and used as input features for each of the models. In addition to this, a target column was created using the following day’s closing price; if the following day's closing price was greater than the current day, target would be assigned 1, and if this wasn’t the case, target would be assigned 0. This target column acts as an expected prediction for all three given models and allows them to be evaluated against their ability to correctly classify this column.

The data was then transformed through the creation of relevant technical indicators that would aid in forecasting GBP/USD price action. A combination of both simple moving averages and indicators that were outlined by Basak.S, 2019 [15] were calculated, the following details how they were calculated.

Simple Moving Averages (SMA) are used to calculate the mean of a range of prices within a rolling window. The window sizes, or periods, used for this project were 2,5,60,250,1000 days respectively. Simple moving averages are calculated as seen in formula (1)

$$\text{SMA} = \frac{A_1 + A_2 + ... + A_n}{n}$$

where:

A_n = the price of an asset at period n

n = the number of total periods

(1)

Relative Strength Index (RSI) is an indicator that aids in assessing if a given asset is overbought or oversold and is an oscillator bound between 0 - 100 values. Traditionally, An asset is usually considered overbought when the RSI is above 70 and oversold when it is below 30 with the standard price lookback being 14 days for the calculation of the indicator. RSI is calculated using formula (2), the average gain and loss would both have a lookback period of 14 days

$$RS = \frac{Avg.Gain}{Avg.Loss}$$

$$RSI = 100 - \frac{100}{1 + RS} \quad (2)$$

Stochastic Oscillator is another momentum indicator like RSI, it returns a percentage by taking a given close price and comparing it to the highest and lowest price of a specified lookback period. Percentages over 80% are considered overbought whilst percentages under 20% are considered oversold. I will be using the standard lookback period which is 14 days. Stochastic Oscillator is calculated using formula(3)

$$\%\text{K} = \left(\frac{\text{C} - \text{L14}}{\text{H14} - \text{L14}}\right) \times 100$$

where:

C = The most recent closing price

L14 = The lowest price traded of the 14 previous trading sessions

H14 = The highest price traded during the same 14-day period

%K = The current value of the stochastic indicator

(3)

Moving Average Convergence Divergence (MACD) is an indicator that models the exponential moving averages (EMA) of two different lookback periods and uses the gap between them to determine upward or

downward momentum. In my case, the two lookback periods used are the 12 and 26 day EMA, MACD also incorporates a signal line which is directly a 9 period lookback of the MACD chart (9 day EMA on the daily chart, 9 hour EMA on the hourly chart etc). The MACD histogram is a visual representation of the difference between the MACD chart and its signal line. This indicator is calculated using formula (4)

$$\text{MACD} = \text{12-Period EMA} \; - \; \text{26-Period EMA} \tag{4}$$

All of the indicator transformations were performed using the functions provided by the pandas-ta library[12]. The transformations caused many of the days to be left with NAN values due to the lookback periods ranging from 2-1000 in the case of certain indicators. These entries were handled by being dropped, causing the start of the training set to be indexed from 2007-10-04 and onwards.

3.3- Model creation

Every model was trained using the same set of data, with this being the first 80% of the preprocessed data frame (2007-10-04 to 2023-03-26), as random subsampling wouldn't make sense given instances where certain days in the test set will be predicted using data from days in the future. Hence, this ordered method of segmenting the data frame was adopted.

The ANN was built using Tensorflow as opposed to Sklearn due to it offering more options as far as configuration was concerned. The hyperparameters that I chose to tune were the **number of nodes in the hidden layer** as well as the **learning rate** of the Adam optimizer. I stuck to using Relu as the activation function over sigmoid as it was not prone to the gradient vanishing problem during backpropagation [6], whereas this was still a relevant issue if sigmoid were used. Binary cross entropy was chosen as the loss function due to the output being a binary value. A random search was performed using the Keras tuner library for 5 epochs, and the configuration of the best hyperparameters was returned based on the set with the highest validation accuracy.

The random forest model was created using Sklearn where the following hyperparameters were considered during the tuning process.
n_estimators: The total number of individual decision trees in the random forest model.
max_depth: The maximum depth each individual decision tree can reach.
min_samples_split: The minimum number of samples required to split a node otherwise, that respective node becomes a leaf.
min_samples_leaf: The minimum number of samples each leaf node requires.
bootstrap: whether bagging is used by the model during training or not.
Much like the ANN model, a randomised search was conducted on the aforementioned hyperparameters and the best combination would then be evaluated.

SVM was also created using Sklearn with the kernel being fixed as RBF the following hyperparameters being tuned via a grid search
c: The regularisation parameter that controls the trade-off between how well fit the model is to the training data vs being a decision boundary

gamma: controls the width of the kernel function used, in my case this would be RBF

Each final model was trained using the training set for 10 epochs to ensure a result representative of their respective performance. The test set consisted of the remaining 20% of entries that were indexed from 2020-03-27 to 2023-05-08.

3.4- Evaluation metrics

In order to evaluate the performance of each model, I will be using **classification accuracy** as one of the metrics to see how each model fared with correctly predicting trends. In addition to this, the models will have their trend predictions used in conjunction with a basic trading strategy during backtesting and the **'Return%**' will act as an additional indicator of performance. Return% indicates the return on investment at the end of the trading period.

However, these two metrics alone aren't holistic representations of how the models performed, hence why precision, recall and f1 score will all be used as they will aid in providing a more comprehensive understanding of model performance.

Precision is a measure of how many "Positive" predictions made by the model are correct, this can be understood using formula (5) and its legend.

TP = True Positive TN = True Negative
FP = False Negative FN = False Negative

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

Recall is a measure of how many positive predictions were correctly identified by the model by also considering the false negatives. This is derived using formula (6).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (6)$$

The **F1 score** is an evaluation metric that combines precision and recall in to a single value thus providing a balanced measure of a models overall performance. It can be calculated using formula (7).

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (7)$$

The aforementioned classification statistics will be provided by the Sklearn classification_report() function. 10 sets of predictions will be made by each model, and the predictions will be averaged and thresholded such that they produce binary outputs. This ensures that the resulting summary statistics are representative of model performance.

3.5- Environment

Backtesting.py was the environment chosen in order to conduct the backtests for the classification models for the reasons stated in the literature review. Backtesting.py defines custom strategies as classes that contain both an init() and next() function. The trading strategy I used is an adaptation of the one provided in the official documentation for Backtesting.py. I established a trade margin of 10:1 and a price delta of 0.004, and leverage was chosen as an option as it is common practise in forex trading. Inside the init() function, each respective model is trained using the training set, and a custom indicator is established in order to hold the predictions for the test set. Inside the next() function, a stop loss and take profit are established that are one price delta away from the close price in each respective direction. Long and short positions are taken at an equivalent of 20% of the account equity if there are no hanging orders.

This backtest was run five times for each model, and mean summary statistics were collated by taking the average of each statistic for the five runs for each model.

4- Results

All three models were trained and tested using the same training sets which employed the same partitioning methods. The support column indicates the amount of samples the set contained of each class, or in the case of the averages, the amount of samples used in order to make the calculations.

	precision	recall	f1-score	support
0	0.52	0.67	0.58	396
1	0.56	0.40	0.47	415
accuracy			0.53	811
macro avg	0.54	0.54	0.53	811
weighted avg	0.54	0.53	0.52	811

Figure 3: Classification statistics Random Forest

	precision	recall	f1-score	support
0	0.49	0.89	0.64	396
1	0.56	0.13	0.21	415
accuracy			0.50	811
macro avg	0.52	0.51	0.42	811
weighted avg	0.53	0.50	0.42	811

Figure 4: Classification statistics SVM

	precision	recall	f1-score	support
0	0.48	0.53	0.50	396
1	0.51	0.47	0.49	415
accuracy			0.50	811
macro avg	0.50	0.50	0.50	811
weighted avg	0.50	0.50	0.50	811

Figure 5: Classification statistics ANN

Starting with precision scores, RF is able to most precisely predict the number of positive downward trends with a score of 52% followed by SVM with a score of 49% and finally ANN with a score of 48%. With regards to the precision of positive upward trends, both SVM and RF are tied at a score of 56% with ANN at 51%.

Interestingly, recall had the most varied results between the models. With SVM's being able to predict the most true positive downward trends among all the real downward trends (TP+FN), with a score of 89% alongside an f1-score of 64% to aid this. Whilst RF and ANN's had downward trend recall scores of 67% and 53% respectively. Due to the models being tested using the same set, this is not a result of an imbalanced dataset but more likely the sensitivity of SVM towards particular features as this may affect how the model captures particular relationships leading it to overpredict a particular trend. This is backed up by its low precision score when compared to its recall for downward trend. ANN had the highest recall when it came to upward trends with a score of 47% whilst RF and SVM had scores of 40% and 13% respectively.

In terms of overall classification accuracy, it is evident from the results above that the random forest model had the best performance when it came to overall prediction accuracy with a score of 53%, where both SVM and ANN models had accuracy scores of 50%. This is a stark difference as during training, SVM performed the best across a 10 fold cross validation with a score of 53% [13]with ANN achieving a score of 49%[14] and RF performing the worst with a score of 47%[15]. The SVM failing to generalise to unseen data may have been a result of overfitting meaning that the regularisation term (C) requires further tuning due to it possibly not making an effective trade-off.

This set of results helped answer two key questions that were established in the onset of this study, the first one being " Which classification model produces the highest accuracy in predicting price trends for GBP/USD?" as well as "Can ensemble classification models increase accuracy of GBP/USD price forecasts compared to individual machine learning models?" The answer to the first question being RF due to it possessing the highest classification accuracy however, upon further evaluation of the supplemental classification statistics, there are areas where the other models succeed over RF; such as SVM when predicting a downward trend. It can also be accepted that within the bounds of this project, ensemble methods (RF) performed better than individual machine learning models (SVM and ANN) with regards to prediction accuracy.

```
Mean Return%: -34.088742130529525
Mean Win Rate%: 26.194996960703328
Mean Final Equity$: 6591.125786947047
Mean Peak Equity$: 10064.600011970806
Mean Max Drawdown%: -34.51118796172749
Mean Average Drawdown%: -24.571697650797887
```

Figure 7: Average Backtesting statistics RF

```
Mean Return%: -45.49105109328755
Mean Win Rate%: 23.862068965517242
Mean Final Equity$: 5450.894890671245
Mean Peak Equity$: 10091.591150636687
Mean Max Drawdown%: -45.98577360788796
Mean Average Drawdown%: -12.34033708179866
```

Figure 8: Average Backtesting statistics SVM

```
Mean Return%: -39.977777340802234
Mean Win Rate%: 23.833707210426734
Mean Final Equity$: 6002.222265919777
Mean Peak Equity$: 10091.301417762948
Mean Max Drawdown%: -41.028882210375336
Mean Average Drawdown%: -22.19056200265594
```

Figure 9: Average Backtesting statistics ANN

The final key question can be answered by taking a look at the backtesting statistics above. Due to the limited robustness of the strategy used, it was unsurprising that all the returns were in the negative. With that being said, the “Buy & Hold Return%” for this period(2020-03-27 to 2023-05-08) was -38.27%. Meaning that the RF model outperformed this strategy in addition to the other models, which was to be expected given that it had the best prediction accuracy out of the three models.

5- Conclusion

Overall, the project successfully accomplished its aims by answering the questions posed in section 1.1. It was observed that the random forest model performed the best as far as classification accuracy was concerned. This was further reinforced by its performance in the backtesting environment when it was used together with the trading strategy, where it produced the highest return%. However, it is important to note that the difference in accuracy between the 3 models is separated by a small margin, leading me to believe that further statistical testing may need to be conducted in order to prove the significance of this result.

It is important to acknowledge that there were a significant number of limitations in this project most notably, the search space and number of fittings used for hyperparameter tuning across all the models. Due to constraints with regards to processing times, random grid searches with small search spaces were used, meaning that there was a major limitation on the amount of hyperparameter configurations tested before selecting the best models.

Due to the lack of feature engineering and ranking, it was hard to interpret the results and see which features contributed the most to specific trends and their reversals. Barring time constraints, I would have liked to explore this further as a means of evaluating various indicators and their effect on a model's ability to accurately forecast trends.

6- Future work

Future work would involve potentially adding another agent responsible for market sentiment analysis due to the nature of the foreign exchange market and how it is also affected by fundamental factors such as geopolitics and economic indicators. This could be achieved through the use of a web scraper to retrieve relevant media, and a language model to process the retrieved data and output a score based on the predicted sentiment of a given piece of media.

I would also like to conduct extensive hyperparameter tuning by incorporating a bigger search space with more configurations for each model's respective hyperparameters, as this may result in a significant increase in model performance. Feature engineering as a means of model interpretability could also be done to categorise the most important indicators in identifying particular price action.

7- Bibliography

[1]Investopedia. (2019). *GBP/USD (British Pound/U.S. Dollar) Definition*. [online] Available at: https://www.investopedia.com/terms/forex/g/gbp-usd-british-pound-us-dollar-currency-pair.asp.

[2]Nielsen, L. (n.d.). *MACHINE LEARNING FOR FOREIGN EXCHANGE RATE FORECASTING*. [online] Available at: https://www.imperial.ac.uk/media/imperial-college/faculty-of-natural-sciences/department-of-mathematics/math-finance/Nielsen_Laurids-Gert_01424460.pdf [Accessed 19 Mar. 2023].

[3]Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, 19(8), pp.1165–1195. doi:https://doi.org/10.1007/s00521-010-0362-z.

[4]Cavalcante, R.C., Brasileiro, R.C., Souza, V.L.F., Nobrega, J.P. and Oliveira, A.L.I. (2016). Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, 55, pp.194–211. doi:https://doi.org/10.1016/j.eswa.2016.02.006.

[5]Guresen, E., Kayakutlu, G. and Daim, T.U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, [online] 38(8), pp.10389–10397. doi:https://doi.org/10.1016/j.eswa.2011.02.068.

[6]KDnuggets. (n.d.). *Vanishing Gradient Problem, Explained*. [online] Available at: https://www.kdnuggets.com/2022/02/vanishing-gradient-problem.html#:~:text=When%20there%20are%20more%20layers.

[7] Weng, B., Lu, L., Wang, X., Megahed, F.M. and Martinez, W. (2018). Predicting short-term stock prices using ensemble methods and online data sources. *Expert Systems with Applications*, 112, pp.258–273. doi:https://doi.org/10.1016/j.eswa.2018.06.016.

[8] Segal, M.R. (2003). *Machine Learning Benchmarks and Random Forest Regression*. [online] Escholarship.org. Available at: https://escholarship.org/uc/item/35x3v9t4.

[9]Huang, W., Nakamori, Y. and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10), pp.2513–2522. doi:https://doi.org/10.1016/j.cor.2004.03.016.

[10] finance.yahoo.com. (n.d.). *GBP/USD (GBPUSD=X) Live Rate, Chart & News - Yahoo Finance*. [online] Available at: https://finance.yahoo.com/quote/GBPUSD%3DX?p=GBPUSD%3DX.

[11]Basak, S., Kar, S., Saha, S., Khaidem, L. and Dey, S., (2019). Predicting the direction of stock market prices using tree-based classifiers. The North American Journal of Economics and Finance, 47, pp.552-567.

[12]Johnson, K. (2023). *Pandas TA - A Technical Analysis Library in Python 3*. [online] GitHub. Available at: https://github.com/twopirllc/pandas-ta#candles-64 [Accessed 8 May 2023].

[13] SVM 10 fold cross validation

```
n_folds = 10
best_model = grid.best_estimator_
cv_error = np.average(cross_val_score(best_model, x_train, y_train, cv=n_folds))
print('The {}-fold cross-validation accuracy score for this classifier is {:.2f}'.format(n_folds, cv_error))
```

```
The 10-fold cross-validation accuracy score for this classifier is 0.53
```

[14] ANN 10 fold cross validation

```
n_folds = 10
history = best_model.fit(x_train, y_train, epochs = n_folds, validation_split=0.2, verbose = 0)
val_acc_per_epoch = history.history['val_accuracy']
cv_error = np.average(val_acc_per_epoch)
print('The {}-fold cross-validation accuracy score for this classifier is {:.2f}'.format(n_folds, cv_error))
```

```
The 10-fold cross-validation accuracy score for this classifier is 0.49
```

[15] RF 10 fold cross validation

```
#Average classification accuracy across 10 folds
n_folds = 10
best_model = rf_random.best_estimator_
cv_error = np.average(cross_val_score(best_model, x_train, y_train, cv=n_folds))
print('The {}-fold cross-validation accuracy score for this classifier is {:.2f}'.format(n_folds, cv_error))
```

```
The 10-fold cross-validation accuracy score for this classifier is 0.47
```