IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 3, 2019, accepted September 18, 2019, date of publication October 7, 2019, date of current version October 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945907

Cost-Sensitive Prediction of Stock Price Direction: Selection of Technical Indicators

YAZEED ALSUBAIE, KHALIL EL HINDI, AND HUSSAIN ALSALMAN
Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Khalil El Hindi (khindi@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research and RSSU, King Saud University through the Research Group under Grant RG-1439-035.

ABSTRACT Stock market forecasting using technical indicators (TIs) is widely applied by investors and researchers. Using a minimal number of input features is crucial for successful prediction. However, there is no consensus about what constitutes a suitable collection of TIs. The choice of TIs suitable for a given forecasting model remains an area of active research. This study presents a detailed investigation of the selection of a minimal number of relevant TIs with the aim of increasing accuracy, reducing misclassification cost, and improving investment return. Fifty widely used TIs were ranked using five different feature selection methods. Experiments were conducted using nine classifiers, with several feature selection methods and various alternatives for the number of TIs. A proposed cost-sensitive fine-tuned naïve Bayes classifier managed to achieve better overall investment performance than other classifiers. Experiments were conducted on datasets consisting of daily time series of 99 stocks and the TASI market index.

INDEX TERMS Cost-sensitive, feature selection, machine learning, market trend, prediction, stock market, technical indicators.

I. INTRODUCTION

Prediction of stock market trends has long been an attractive topic that has been extensively studied by researchers from various fields, because of its importance in making informed investment decisions and its potential financial benefits. Investment strategies can always benefit from new insights and tools that better utilize historical data to predict the future.

A wide range of approaches is used to analyze and forecast market behavior. They include fundamental and technical analysis, statistical and multiple criteria decision aiding (MCDA), textual analysis, data mining, and soft computing. Machine learning (ML) algorithms are increasingly being used in the financial sector [1]. ML has the potential of identifying stock trends from massive amounts of data that capture the underlying stock price dynamics.

In computational finance, the forecasting of financial markets using ML approaches often relies on technical indicators (TIs) as the attributes, or features, of the input datasets. However, researchers have not reached a consensus about a suitable collection of TIs to use. The choice of TIs suitable for a given forecasting model remains an area of active research. Some researchers use 10 TIs, whereas others use 20 or more [2]–[5]. Numerous TIs are being used by investors, and their number is still growing, as a result of the addition of new indicators and variants.

This work focuses on the selection of a minimal number of relevant TIs as input data, to reduce the dimensionality of the forecasting problem while achieving acceptable prediction performance. Using a minimal number of input features is crucial for successful stock market forecasting [6]. Each input feature introduces an additional dimension, which increases the sparsity of training data in the data space; thus, the amount of training data required increases exponentially with the number of features, to cover each combination of feature values.

Evaluating which TIs are relevant, and how many TIs to use, is accomplished by experimenting with several feature selection configurations on different prediction models. A ranking of 50 widely used TIs is developed, using three different feature selection methods and two wrapper methods:

The associate editor coordinating the review of this manuscript and approving it for publication was Changsheng Li.

one accuracy-based and the other cost-based. The ranking is presented in the form of heat maps, which can serve as a guide for selecting TI

Experiments investigate the impact of several feature selection configurations on the performance of various prediction models, using three metrics: classification accuracy, misclassification cost, and the expected investment return of a trading system simulation. The research utilizes nine classifiers: naïve Bayes (NB), fine-tuned naïve Bayes (FTNB), artificial neural network (ANN), support vector machine (SVM), MetaCost wrapper with these four classifiers, and cost-sensitive fine-tuned naïve Bayes (CSFTNB). Using different ML algorithms is necessary to neutralize the effect of their inductive bias.

The last five classifiers listed above are cost-sensitive classifiers. Cost-sensitive techniques are essential to deal with imbalanced datasets, such as those generated from the time series of stock daily trading data. These datasets usually have an unequal distribution between their classes, as shown in TABLE 3. In such cases, classifiers tend to be biased toward the majority class. That is, they tend to wrongly classify an instance from a minority class as belonging to a majority class, even though minority classes are more important in making predictions and investment decisions.

CSFTNB is proposed in this research, to make the NB classifier cost sensitive. CSFTNB aims to fine-tune the probabilities so that NB is more likely to make lower-cost predictions. Once the classifier is tuned, new instances are classified in the same manner as with the NB classifier.

The results show that choosing a large number of TIs (above 30) is likely to reduce performance, with respect to all three performance metrics. More specifically, results show that the highest accuracy is achieved with at least 10 TIs, the lowest cost with five TIs, and the best investment performance with five to 10 TIs. The research finds that accuracy, and those achieving the lowest cost, failed to produce the best performance in a trading simulation. However, a cost-sensitive fine-tuning of an NB classifier achieved the highest balance between investment return and risk.

It is worth mentioning that numerous studies have been conducted using ML to predict the direction of movement of the world's major stock markets. A few of them, such as [3], [7], [8], have been applied to the Middle Eastern markets. Saudi Arabia is a G20 member and plays a significant role in the global economy. Its economy is the largest in the Middle East and North Africa (MENA) region, according to the International Monetary Fund (IMF) in its World Economic Outlook for 2015 [9]. Hence, the decision was taken to base this work on data obtained from the Saudi stock market.

II. RELATED WORK

There are two major approaches to financial time series prediction: statistical models and ML approaches. Statistical methods generally assume a linear process while modeling the generation of the underlying time series and predicting its future values. However, financial time series are inherently complex, highly noisy, dynamic, nonlinear, nonparametric, and chaotic [10], [11].

Various ML techniques, such as ANN, SVM, NB, and genetic algorithms [11]–[14], have been applied to stock market prediction. Support vector regression (SVR) models have been used to forecast various nonlinear and chaotic time series, such as in [15]–[17], and have also been successfully applied to stock market forecasting [18]. The expanding literature on the use of ML techniques for sentiment and textual analysis in finance was reviewed in [14]. News, and other company information, was classified and used to generate sentiment series. The study compared the classification performance of the lexicon-based and ML approaches: specifically ANN, SVM, and NB. Big data analysis techniques were used in [19] for sentiment analysis, to forecast stock price movement. Ou and Wang [20] applied 10 different ML techniques to predict the price movement of the Hang Seng index of the Hong Kong stock market. A benchmark of ensemble methods (random forest, AdaBoost, and kernel factory) against single-classifier models (ANN, logistic regression, SVM, and k-nearest neighbors) was reported in [21].

Cost-sensitive ML addresses problems in which different classification errors have different costs. There are numerous practical applications in which the costs of different errors are not equal. For example, in medical diagnosis, it is much less costly to wrongly diagnose a healthy person as sick, and incur further expense for more medical tests, than to misdiagnose an ill person as healthy and risk a loss of life. Similarly, there are varying costs associated with wrongly predicting the price trend direction as up, down, or flat. A cost matrix is usually built, based on the number of defined classes for a given problem domain.

One of the popular methods for cost-sensitive learning is rescaling [22]. Rescaling can be achieved in different ways, such as by sampling the training instances proportionally to their cost. Sampling pushes the decision boundaries away from the classes with high cost [22]. There are several approaches to sampling, such as oversampling, undersampling, cloning, and instance weighting [23]. All of these techniques work by changing the class distribution, by assuming that it does not obey the underlying real distribution. This is done either by adding instances from the minority class or removing instances from the majority class. Chawla proposed the synthetic minority oversampling technique (SMOTE) [24]. Jiang introduced the minority cloning technique (MCT) [25], for cloning each minority class instance several times, based on its similarity to a virtual minority mode instance.

Instance weighting is a sampling technique that assigns a normalized weight to each instance, which is based on the cost of misclassification. This technique was used in [26] to propose cost-sensitive Bayesian network classifiers. The weighting method works whenever the underlying cost-blind classifier can accept instance weights directly [23].

Thresholding [27], [28] is another method for cost-sensitive learning. In this method, a threshold is used to

classify instances. This method works when the underlying cost-blind classifier can estimate the class probabilities. The MetaCost technique, discussed in detail in the next section, uses thresholding to relabel the training instances based on the cost of misclassification.

Both sampling and thresholding fall into the meta-learning category of cost-sensitive classification. Meta-learning works by designing a wrapper around the underlying cost-blind classifier, without changing it. The wrapper generally alters the training data, to transform a classifier to make it cost-sensitive. The other category is cost-sensitive learning, in which the learner is modified to make the classifier cost-sensitive.

A. METACOST WRAPPER

Domingos [27] proposed the MetaCost method, which is a wrapper procedure for making classifiers cost-sensitive using a cost-minimizing procedure. The procedure treats the underlying classifier as a "black box": it has no knowledge of how the classifier functions and does not change it. It uses an ensemble of classifiers, similar to the bagging method [29], to relabel each training instance with a label (class) that minimizes the expected cost; this class may differ from the instance's actual class. The classifier is then trained using the relabeled data; it is trained to predict the class with the lowest cost, rather than the correct class.

The MetaCost procedure works best for unstable learners: that is, in situations where a slight change in the training set produces a substantially different classifier. Therefore, it is not expected to work well for NB classifiers, because NB is a stable learning algorithm [30]. The procedure works as follows [27]:

- An ensemble of the underlying classifier is used to estimate the class probabilities for each training instance.
- The class with the lowest expected misclassification cost, or conditional risk $R(i \mid x)$, is selected.

$$R(i \mid x) = \sum p(j \mid x)\, C(i, j) \tag{1}$$

where $C(i, j)$ is the cost of classifying the instance to be of class i when in reality it belongs to class j, and $p(j \mid x)$ is the probability of class j, given the instance x.

- The class with minimal conditional risk is chosen to relabel the training instance.
- At the final stage, a classifier is trained with the relabeled training dataset.

B. FINE-TUNING THE NB

El Hindi [31] proposed an algorithm to improve the classification accuracy of the NB algorithm by fine-tuning the probability values used by NB, to find better estimates for these values. The method, called FTNB, showed a significant improvement in classification accuracy compared to the NB algorithm. In [32], a slightly different method was proposed to fine-tune Bayesian networks.

The FTNB algorithm comprises two learning stages. In the first stage, the training set is used to construct a classical NB classifier, and the training set is used in the usual way to estimate the required probability terms. In the second stage, the training set is used again to find better estimates for the probability terms of NB. If the NB classifier mistakenly classifies a training instance, it means that the predicted class, $C_{predicted}$, has a greater computed probability than the instance's actual class (correct class), C_{actual}, given the instance's attribute values. Therefore, the algorithm increases the values of the probability terms involved in computing the probability of the actual class and decreases those of the terms involved in computing the probability of the predicted class. Specifically, it increases the values of the probability terms $p(a_i \mid C_{actual})$ for each attribute value a_i. Additionally, the algorithm decreases the probability of the mistakenly predicted class, $C_{predicted}$, by decreasing the values of the probability terms that contributed to this error, namely, the terms $p\left(a_i \mid C_{predicted}\right)$, for each attribute value a_i. FTNB neither increases $p(C_{actual})$ nor decreases $p\left(C_{predicted}\right)$ because it was found empirically that fine-tuning these terms has no effect on the performance of NB [31].

The size of the update, δ_i, is proportional to the size of the error, which is computed as

$$Error = \left|p(c_{actual} \mid a_1, a_2, \ldots, a_n) - p\left(c_{predicted} \mid a_1, a_2, \ldots, a_n\right)\right| \tag{2}$$

where $P(c_o \mid a_1, a_2, \ldots, a_n)$ is the probability of class c_o given the attributes $a_1, a_2, \ldots, a_n$. It is calculated using the equation

$$p(c_o \mid a_1, a_2, \ldots, a_n) = p(c_o) \cdot \prod_{i=1}^{n} p(a_i \mid c_o) \tag{3}$$

where n is the number of attributes in a given instance and $p(c_o)$ is the frequency probability of class c_o in the dataset.

When computing $\delta_{(t+1)}(a_i, c_{actual})$, the size of the update should be large for small probability values and small for large probability values. The reason is that small probability values are more likely to be responsible for misclassification than large probability values. This goal is achieved by adjusting the size of the update to be proportional to

$$\alpha \cdot p(max_i \mid c_{actual}) - p(a_i \mid c_{actual}) \tag{4}$$

where max_i is the value of the i^{th} attribute with the maximum probability, given c_{actual}. This equation ensures that a greater difference between $p(a_i \mid c_{actual})$ and $p(max_i \mid c_{actual})$ leads to a larger update step. α is a constant greater than or equal to 1, and is used to adjust the size of the update step for the term $p(a_i \mid c_{actual})$, relative to its distance from $p(max_i \mid c_{actual})$. If $\alpha$is set to 1, the size of the update step for $p(max_i \mid c_{actual})$ is 0, whereas, if it is greater than 1, the size of the update step for $p(a_i \mid c_{actual})$ is greater than 0. The following equation considers all factors mentioned above to determine the size of the update step

$$\delta_{t+1}(a_i, c_{actual}) = \eta \cdot (\alpha \cdot p(max_i|c_{actual}) - p(a_i|c_{actual})) \cdot error \tag{5}$$

where η is a constant between 0 and 1, and determines the learning rate.

To compute $\delta_{t+1}\left(a_i, c_{predicted}\right)$, the probability terms (responsible for $c_{predicted}$ having a higher probability than c_{actual}) need to be decreased. Furthermore, the size of the decrement step needs to be proportional to the size of the error. This is achieved by making the size of the update proportional to

$$\beta \cdot p\left(a_i \mid c_{predicted}\right) - p\left(min_i \mid c_{predicted}\right) \tag{6}$$

where min_i is the value of the i^{th} attribute with the minimum probability, given $c_{predicted}$. β is a constant that is greater than or equal to 1, and is used to control the size of the update step by scaling the term $p\left(a_i \mid c_{predicted}\right)$ relative to its difference from $p\left(min_i \mid c_{predicted}\right)$. A larger value of β results in a larger update step. The update step size is defined by the equation

$$\delta_{t+1}\left(a_i, c_{predicted}\right) = \eta \cdot (\beta \cdot p(a_i|c_{predicted}) - p(min_i|c_{predicted})) \cdot error \tag{7}$$

III. PROBLEM STATEMENT AND APPROACH

In this study, trend prediction is considered as a classification problem, with three class values (labels): up, down, and unchanged. These three labels represent the trend direction d_i, where $d_i \in \{d_1 = Up, d_2 = Flat, d_3 = Down\}$. For each trading day, a vector of attributes is calculated, representing the values of several TIs in addition to the daily closing price. These daily data instances and labels (for each stock) are used to generate the datasets for 100 stocks from the Saudi stock market.

Numerous TIs are being used by investors. The number of indicators is still growing, as a consequence of the addition of new indicators and variants that aim to achieve better results. The usefulness of an indicator may vary between different stocks and markets; therefore, deciding which indicators to use is not an easy task. Using TIs is more of an art than a science. Some powerful insights into profitable technical patterns and strategies are discussed in [33].

This research investigates how the performance of a stock market prediction system changes according to the number of input TIs. The performance is measured using three metrics: accuracy, cost of misclassification, and Sharpe ratio. The experiments are conducted using several filtering and classification methods, as explained in the following paragraphs of this section.

The underlying classification problem in this work is to build a classifier that takes the closing price of a certain period and the values of the selected TIs in that period, and then predicts the direction of the trend in the next period. The training data consist of instances of the form $< x_i, d_i >$, where x_i is a vector of n attributes $a_1, a_2, \ldots, a_n$, and each vector x_i is labeled as $d_i \in \{d_1 = Up, d_2 = Flat, d_3 = Down\}$.

The study uses nine classifiers, five of which are cost-sensitive. The classifiers are NB, FTNB, ANN, SVM, MetaCost wrapper with NB (M-NB), MetaCost wrapper with FTNB (M-FTNB), MetaCost wrapper with ANN (M-ANN), MetaCost wrapper with SVM (M-SVM), and the newly proposed CSFTNB.

Each classifier's performance is evaluated and compared, using the classification accuracy metric and the misclassification cost metric. As the third performance metric, daily predictions are generated and used as input to a trading simulator, to calculate the expected investment return.

The experiments start with the use of 50 widely used TIs, listed in TABLE 1, and then apply three feature selection methods—namely, GainRatio, ReliefF, and Correlation—in addition to two NB wrapper methods: one accuracy-based and the other cost-based. These five feature selection methods are used to rank the TIs for each dataset. The nine classifiers are trained and tested with distinct groups of datasets, containing different numbers of highest-ranking TIs, to record the performance levels for both accuracy and misclassification cost. For each of these distinct combinations of filter type, ranking level, and classifier method, daily predictions

TABLE 1. Technical indicators.

#	Technical Indicator	Indicator Type	Description
1	Close	General	Return % of Closing Price
2	AD	Volume	Chaikin A/D Line
3	ADOsc	Volume	Chaikin A/D Oscillator
4	ADX	Momentum	Average Directional Movement Index
5	ADXR	Momentum	ADX Index Rating
6	APO	Momentum	Absolute Price Oscillator
7	AR_UP	Momentum	Aroon Up
8	AR_DN	Momentum	Aroon Down
9	ARO	Momentum	Aroon Oscillator
10	ATR	Volatility	Average True Range
11	BBands_UP	General	Bollinger Upper Line
12	BBands_Mid	General	Bollinger Midline
13	BBands_LO	General	Bollinger Lower Line
14	BOP	Momentum	Balance Of Power
15	CCI	Momentum	Commodity Channel Index
16	CMO	Momentum	Chande Momentum Oscillator
17	DX	Momentum	Directional Movement Index
18	DEMA	General	Double Exponential Moving Average
19	KAMA	General	Kaufman Adaptive Moving Average
20	HT_DCPERIOD	Cycle Indicator	Hilbert Transform - Dominant Cycle Period
21	HT_DCPHASE	Cycle Indicator	Hilbert Transform - Dominant Cycle Phase
22	HT_Trendline	Cycle Indicator	Hilbert Transform - Trend vs Cycle
23	MACD	Momentum	Moving Average Convergence/Divergence
24	MAX	Statistical	Max Close
25	MFI	Momentum	Money Flow Index
26	Midpoint	General	MidPoint over period
27	Midprice	General	Midpoint Price over period
28	MIN	Statistical	Min Close
29	MOM	Momentum	Momentum
30	NATR	Volatility	Normalized Average True Range
31	OBV	Volume	On Balance Volume
32	PPO	Momentum	Percentage Price Oscillator
33	ROC	Momentum	Rate of change
34	RSI	Momentum	Relative Strength Index
35	SMA	General	Simple Moving Average
36	STD	Statistical	Standard Deviation
37	Stoch_SK	Momentum	Stochastic Slow %K
38	Stoch_SD	Momentum	Stochastic Slow %D
39	StochF_FK	Momentum	Stochastic Fast %K
40	StochF_FD	Momentum	Stochastic Fast %D
41	StochRSI_FK	Momentum	Stochastic Relative Strength Index Fast %K
42	StochRSI_FD	Momentum	Stochastic Relative Strength Index Fast %D
43	T3	General	Triple Exponential Moving Average
44	TRIMA	General	Triangular Moving Average
45	TRIX	Momentum	1-day ROC of a Triple Smooth EMA
46	TP	Price Transform	Typical Price
47	ULTOSC	Momentum	Ultimate Oscillator
48	WCLPRICE	Price Transform	Weighted Close Price
49	WillR	Momentum	Williams' %R
50	WMA	General	Weighted Moving Average

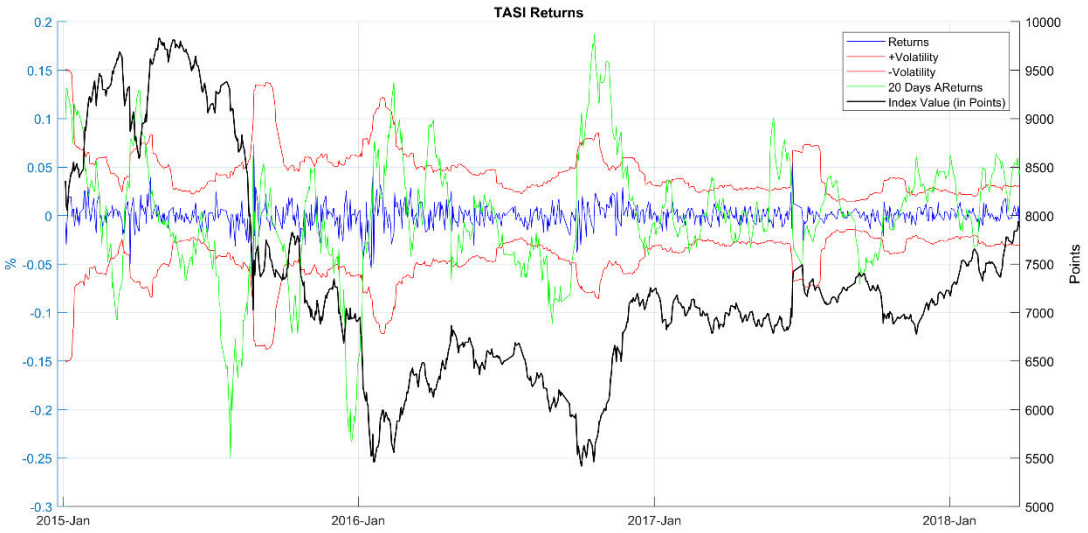

FIGURE 1. Class labeling. The threshold band [−V, +V] is used to determine the class label of each instance. This method is used to label the 100 datasets generated from computing the selected TIs for each stock over the three-year period.

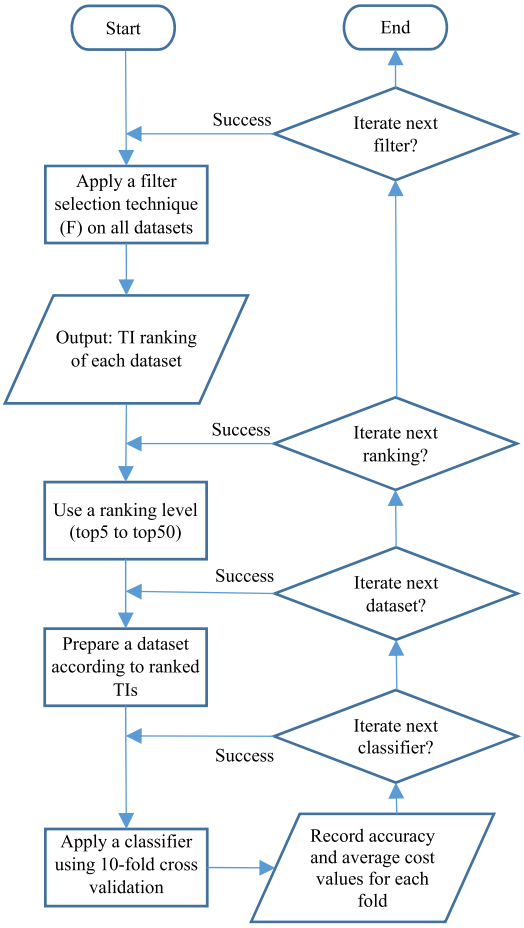

FIGURE 2. High-level flowchart for feature selection and classification.

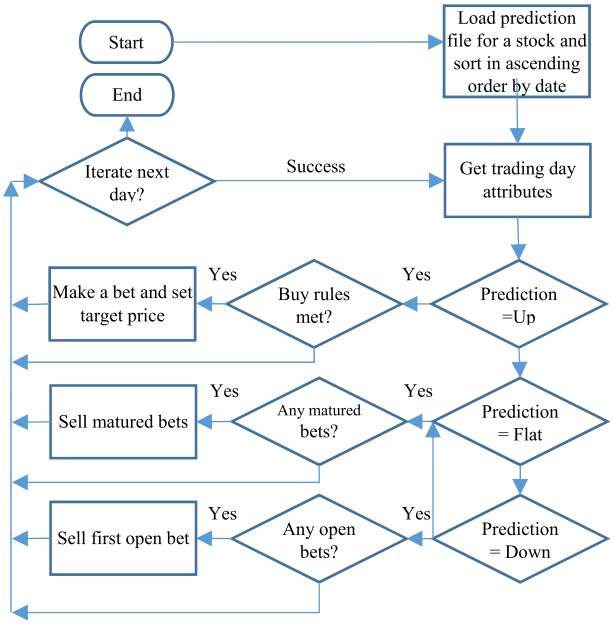

FIGURE 3. Trading strategy flowchart.

are generated, to be used later as input to a trading system simulator. The simulator applies a straightforward trading strategy to calculate the investment return percentage, winning rate, and Sharpe ratio. The procedure for applying the filters and generating daily predictions is depicted in Figure 2, and the trading strategy in Figure 3.

A. FEATURE SELECTION

One of the phenomena often encountered in ML is the ''curse of dimensionality,'' as referred to by Bellman [34]. This means that, with a given level of accuracy, the number of training samples needed to estimate a target function grows exponentially with the number of variables or features (i.e., dimensions) that it comprises. Each additional dimension increases the sparsity of training data in the data space, and thus, the requirement for more training data increases exponentially, to cover all combinations of feature values.

Generally, there are two types of feature selection: feature extraction, and feature subset selection. Feature extraction mechanisms transform the set of features into

a lower-dimensional feature space; such methods include principal component analysis (PCA) and linear discriminant analysis (LDA). In contrast, the feature selection process aims to find relevant features that make the greatest contribution to the prediction desired output. Feature selection methods are further divided into three groups: filter methods, wrapper methods, and embedded methods. This study applies filter methods and wrapper methods, to find the minimal number of relevant TIs to reduce the dimensionality of the forecasting problem while achieving acceptable prediction performance.

1) FILTERS

Filter methods are computationally fast because they require no learning. Filters are mostly univariate: they apply a performance test on each feature independently to determine its relevance to the output class. The filtering result is usually a list of features ranked according to their relevance. Typical performance measures used for feature filtering include information, distance, and statistical measures [35]. The three filters used in this work are:

- GainRatio: a modification of information gain, developed to reduce its bias toward features with large numbers of possible values [36].
- Correlation: a statistical filter.
- ReliefF: a distance filter.

Reference [35] provides a good overview of filter methods.

2) WRAPPERS

Wrapper methods use a learning algorithm (a classifier) to evaluate features and decide which features to retain. Wrappers evaluate various subsets of features, based on the underlying classifier performance. For subset generation, they use a search strategy, such as the forward selection and backward elimination strategies [37]. Forward selection starts with an empty set, and then adds the most relevant feature at each iteration until the stop condition is satisfied. Backward elimination starts with a set of all features, and then eliminates the least relevant feature at each iteration until the stop condition is satisfied. Stepwise strategies vary, by combining elements of forward selection and backward elimination, in some fashion.

This research uses Weka's greedy stepwise search set to conservative forward selection [38], and NB as the underlying classifier. The default evaluation measure in Weka is accuracy. Cost is not one of the available evaluation measures. Therefore, a cost-sensitive subset evaluation wrapper was developed, which included average cost as an evaluation measure.

IV. COST-SENSITIVE FINE-TUNING OF NB

This research proposes CSFTNB, a method for cost-sensitive fine-tuning of the naïve Bayesian classifier. CSFTNB aims to fine-tune the probabilities so that NB gives the lower-cost class. It differs from Domingos' MetaCost method [27] in the sense that it does not relabel data, but modifies the probabilities to make the lower-cost class more likely to be produced as the output of the NB classifier.

Algorithm 1 CSFTNB (Training Instances)

```
Input:
Cost matrix C; where C(c_pred, c_targ) is the cost of
predicting the class c_pred, whereas, in fact, it is c_targ
Training instances
Phase 1:
Use the training instances to estimate the value of each
probability term used by the NB algorithm.
Phase 2:
t = 0
Do
    For each training instance x, do
        Let c_pred = classify(x), the predicted class;
        Let c_targ = the target class with lowest
        prediction cost;
        if c_pred <> c_targ then
            Compute the misclassification cost as
            ErrorCost = C(c_pred, c_targ) * |P(c_targ|x)) −
            P(c_pred|x))|;
            For each attribute value a_k of x do
                Compute δ_(t+1) (a_k, c_targ);
                P_(t+1) (a_k | c_targ) =
                P_t (a_k | c_targ) + δ_(t+1) (a_k, c_targ);
                Compute δ_(t+1) (a_k, c_pred);
                P_(t+1) (a_k | c_pred) =
                P_t (a_k | c_pred) + δ_(t+1) (a_k, c_pred);
            End
        End
    End
    t = t + 1;
While total conditional risk ∑ R (c_pred | x) is
decreasing;
```

Like FTNB, the CSFTNB algorithm consists of two phases. In the first phase, the training set is used to construct a classical NB classifier, and it is used in the usual way to estimate the required probability terms. The second phase is the cost-sensitive fine-tuning of the probability values. Its purpose is to fine-tune the probability values to reduce the cost of misclassification. The probability values are fine-tuned in such a manner that the NB classifier is more likely to produce the lowest-cost class for the given instance. The aim is to gradually reduce the conditional risk, $R\left(c_{pred} \mid x\right)$, of the predicted class c_{pred}, given an instance, x.

The CSFTNB algorithm described in Algorithm 1 modifies the probability values so that the NB classifier produces the lowest-cost class c_{targ}, which may not be the same as the actual class of the training instance. c_{targ} is calculated using (8); hence, depending on the cost matrix, it may differ from c_{pred}. There are several differences between CSFTNB and FTNB: they use different error functions, tuning equations, and termination conditions. However, once the

classifier is tuned, it is used to classify new instances in the same manner as the NB classifier.

In the fine-tuning stage, each instance in the training set is classified and compared with the target class. If the predicted class c_{pred} is not the lowest-cost class c_{targ}, then the algorithm makes gradual updates to find better estimates for the probability terms. The probability values $P\left(a_k \mid c_{targ}\right)$ need to be gradually increased for each attribute value a_k to make the NB classifier more likely to produce the lowest-cost class c_{targ}, whereas the probability values $P\left(a_k \mid c_{pred}\right)$ need to be gradually decreased for each attribute value a_k. The lowest-cost class is defined as

$$\underset{v \in class\ labels}{argmin} \quad P\left(v \mid a_1, a_2, \ldots, a_n\right) C\left(v, c_{targ}\right) \tag{8}$$

The step size δ_{t+1} of the update should be proportional to:

- The cost of predicting class c_{pred} when, in fact, the lowest-cost class is c_{targ}; i.e., $C\left(c_{pred}, c_{targ}\right)$.
- The difference between the probabilities of the predicted and lowest-cost classes: i.e., $\left|P\left(c_{pred} \mid a_1, a_2, \ldots, a_n\right) - P\left(c_{targ} \mid a_1, a_2, \ldots, a_n\right)\right|$. The greater the difference, the larger the update step.

These two elements are combined to calculate the cost of the error as

$$\begin{aligned} ErrorCost = {} & C\left(c_{pred}, c_{targ}\right) \\ & * \left|P\left(c_{targ} \mid a_1, a_2, \ldots, a_n\right)\right. \\ & \left. - P\left(c_{pred} \mid a_1, a_2, \ldots, a_n\right)\right| \end{aligned} \tag{9}$$

The update step size δ_{t+1} is computed iteratively for each probability term and added to the previous term's value. The update value is either positive, to increase $P\left(a_k \mid c_{targ}\right)$, or negative, to decrease $P\left(a_k \mid c_{pred}\right)$.

Equations (10) and (11) define the update step size for c_{targ} and c_{pred}. They differ from the corresponding equations of FTNB in two ways. First, the algorithm uses *ErrorCost*, defined in (9), instead of *error*, defined in (2). Second, it uses a decaying learning rate η, to make the update process more gradual as the number of iterations t increases.

$$\begin{aligned} \delta_{t+1}\left(a_i, c_{targ}\right) = {} & \frac{\eta}{10^t} \cdot (\alpha \cdot p(max_i|c_{targ}) \\ & - p(a_i|c_{targ})) \cdot ErrorCost \end{aligned} \tag{10}$$

$$\begin{aligned} \delta_{t+1}\left(a_i, c_{pred}\right) = {} & \frac{\eta}{10^t} \cdot (\beta \cdot p\left(a_i \mid c_{pred}\right) \\ & - p(min_i|c_{pred})) \cdot ErrorCost \end{aligned} \tag{11}$$

CSFTNB also differs from FTNB in its termination condition. CSFTNB continues the fine-tuning process until the total conditional risk $\sum R\left(c_{pred} \mid x\right)$ no longer decreases. During each iteration, the algorithm calculates the total conditional risk for the training instances, as defined by (1), and terminates the fine-tuning process once this total has ceased to decrease.

Because CSFTNB gradually increases the likelihood of predicting the lowest-cost class, it decreases the conditional risk for each training instance, and thereby the total conditional risk. Once CSFTNB is fine-tuned, it can be used to classify new instances in the same manner as an NB or FTNB classifier.

V. EMPIRICAL EXPERIMENT

The Saudi financial market's daily historical data is available on several commercial websites by subscription. Each data point consists of the following attributes: Date, Open, High, Low, Close, Change, Value, and Volume. Each stock has a unique ticker code. The data for 99 companies, and the TASI market index, were used in the experiments, to construct 100 datasets. These companies are large to medium-sized. They are well-known organizations in banking, telecommunications, utility, and various other market segments. The experiments used data for the period from January 1, 2015 to March 31, 2018 for each of the 100 datasets. These datasets, including a summary of the class categories, are listed in TABLE 3. Each dataset was subjected to a preprocessing phase, wherein each trading day's data was transformed to a feature vector of TIs (an instance). Each instance has a class value that indicates the direction of the stock price: up, flat, or down.

A. PREPARING THE DATASETS

This section describes the input feature set and the preprocessing work, as well as the class labeling. Matlab was used for dataset preparation, labeling, and normalization.

1) PREPROCESSING THE FEATURES

Several TIs were evaluated to be used as input features in addition to the closing price (as a return percentage). Fifty of the most common TIs were selected, as described in TABLE 1. The TIs were calculated using the Technical Analysis Library (TA-Lib) [39] (available at www.ta-lib.org), which is an open-source library widely used by commercial, private, and open-source applications for technical analysis. Because the Java version of TA-Lib is not directly usable from within Matlab, we developed a Java wrapper to facilitate parameter passing between the two frameworks. The TIs were calculated from the historical trading time series, including open, close, high, and low prices, and trading volume. A medium-range lookback window w, of the past 20 trading days, was chosen to calculate the TIs. The prediction window was set to the same length. The best prediction performance is expected when these two windows are equal as reported in [2]. The features were then normalized to have zero mean and unit variance.

2) CLASS LABEL CALCULATION

The class labeling process used two variables to label each instance: the past price volatility V and the future stock accumulated returns *AReturns*. The calculation of these two variables was based on a specific period (or window) whose size was set to be $w = 20$, which is similar to the window size used to calculate the TIs. Stock price volatility is defined as the standard deviation of the daily percentage change, for a given period. Volatility is usually annualized by multiplying

TABLE 2. Cost matrix.

Cost Matrix $C(i,j)$		Actual Class (j)		
		Up	Flat	Down
Predicted Class (i)	Up	0	$\propto \cdot \frac{P(Up)}{P(Flat)}$	$\propto \cdot \frac{P(Up)}{P(Down)}$
	Flat	$\propto \cdot \frac{P(Flat)}{P(Up)}$	0	$\propto \cdot \frac{P(Flat)}{P(Down)}$
	Down	$\propto \cdot \frac{P(Down)}{P(Up)}$	$\propto \cdot \frac{P(Down)}{P(Flat)}$	0

by the square root of the number of trading days in a year; therefore, for the period of w trading days, the standard deviation was multiplied by the square root of w, as in (12). The class labeling process used the volatility to create a threshold band, by using the positive and negative values of V. This threshold band was then used to evaluate the accumulated returns, *AReturns*. *AReturns* is simply the sum of the daily percentage change in price for the next w trading days.

$$V = STD\,(Past\ w\ daily\ returns) * \sqrt{w} \tag{12}$$

The class labeling process labeled each instance following the rules:

- If *AReturns* exceeds the upper threshold, $+V$, then the class of that day is labeled Up.
- If AReturns is below the lower threshold, $-V$, then the class is labeled Down.
- If *AReturns* is within the threshold band, then it is labeled Flat.

$$ClassLabel = \begin{cases} Up; & AReturns > +V \\ Flat; & -V \leq AReturns \geq +V \\ Down; & AReturns < -V \end{cases} \tag{13}$$

Figure 1 shows the threshold band between the two red lines: the upper and lower lines indicate the positive volatility $+V$ and negative volatility $-V$, respectively, of the TASI returns. The blue line indicates the daily TASI returns, and the green line indicates the *AReturns* of the next w days after each trading day. These lines are used for labeling each data point in the datasets. When the green line crosses above the upper red line, that period is labeled Up. The period in which the green line is below the red line is labeled Down. The remaining periods, in which the green line remains between the two red lines, are labeled Flat.

3) DISCRETIZATION

This research used (*weka.filters.unsupervised.attribute*), a Weka package, for discretizing all numeric attributes and replacing the missing values. This package, and the methods used, are explained in the Weka document [38].

B. EVALUATING THE PERFORMANCE OF THE CLASSIFIERS

A prediction system was constructed with the architecture shown in Figure 2. The system was trained and tested separately for each dataset's distinct group of TIs, resulting from each feature selection method. These distinct groups are the ranked lists of either the top five, top 10, top 20, top 40, or all 50 TIs, in the case of filters or the generated subset of TIs from the wrappers. Ten-fold cross-validation was applied in all experiments, to calculate the accuracy and misclassification cost. The Weka Java framework was used to implement all feature selection techniques and classification methods, as depicted in Figure 2. The predictions needed for the simulation were generated similarly, except for the bottom two boxes in the figure; the classifiers were trained based on rolling periods, to generate predictions. These generated predictions were later used in Matlab to simulate a trading strategy for each configuration, from which the investment return percentage, winning (or hit) rate, and Sharpe ratio were calculated.

1) ACCURACY

Classification accuracy was calculated by counting the true positives and dividing by the number of instances. Because there are three classes $\{d_1 = Up, d_2 = Flat, d_3 = Down\}$, the formula is as follows:

$$Accuracy = \frac{True\,(d_1) + True\,(d_2) + True(d_3)}{N} \tag{14}$$

The accuracy values were averaged over all 100 datasets, to produce one accuracy number for each combination of filter method, feature subset size, and classifier.

2) COST OF MISCLASSIFICATION

A 3×3 matrix $C\,(i,j)$, shown in Table 2 was used as the cost matrix. The order of the matrix is 3×3 because there are three distinct classes. Following Domingos [27], a different cost matrix was built for each dataset, based on the class distributions in the training set. The cost of predicting class i, where the actual class is j, was calculated from the class probabilities, such that

$$C\,(i,j) = \alpha \cdot \frac{P\,(i)}{P\,(j)} \tag{15}$$

where α is a constant. This cost model provides the highest misclassification cost while classifying a rare class j as a common class i, and provides the lowest cost when classifying a common class as a rare one. The cost of a correct prediction (i.e., $i = j$) is set to zero. As a result, the diagonal elements of the cost matrix are always equal to zero. It is worth mentioning that the value of the constant α is irrelevant in making comparisons, and was set to 10 in the implementation.

The cost values were averaged over all 100 datasets, as with accuracy.

3) SIMULATION OF A TRADING SYSTEM

The Saudi stock market does not allow for short positions on listed stocks; therefore, only long trades are considered in the simulation. The simulation adopted a straightforward trading strategy, as shown in Figure 3, to test the predictions made by each of the nine classifiers for each distinct configuration.

TABLE 3. 100 generated datasets.

#	Ticker	Short Name	All	Up	Flat	Down	#	Ticker	Short Name	All	Up	Flat	Down
1	T1010	RIBL	2060	415	1302	343	.	.	.	.	.	.	.
2	T1020	BJAZ	2060	455	1245	360	.	.	.	.	.	.	.
3	T1030	SAIB	2060	436	1237	387	.	.	.	.	.	.	.
4	T1050	BSFR	2060	369	1416	275	51	T3050	SPCC	2060	366	1357	337
5	T1060	SABB	2060	379	1382	299	52	T3060	YCC	2060	389	1225	446
6	T1080	ARNB	2060	388	1290	382	53	T3080	EPCCO	2060	348	1240	472
7	T1120	AL RAJHI	2060	459	1317	284	54	T3090	TCC	2060	377	1325	358
8	T1140	ALBILAD	2060	515	1230	315	55	T3091	JOUF CEMENT	1881	386	1120	375
9	T1150	ALINMA	2060	367	1376	317	56	T4001	A.OTHAIM MARKET	2060	469	1357	234
10	T1201	TAKWEEN	1516	277	938	301	57	T4004	DALLAH HEALTH	1303	225	915	163
11	T1210	BCI	2060	344	1366	350	58	T4010	DUR	2060	375	1331	354
12	T1211	MAADEN	2060	475	1302	283	59	T4020	SRECO	2060	419	1285	356
13	T1212	ASTRA INDUST	2060	362	1349	349	60	T4050	SASCO	2060	381	1341	338
14	T1213	ALSORAYAI GROUP	2001	363	1182	456	61	T4070	TAPRCO	1966	418	1271	277
15	T1214	SHAKER	1943	315	1234	394	62	T4080	ASEER	2060	426	1242	392
16	T1330	ALKHODARI	1836	335	1072	429	63	T4090	TAIBA	2060	466	1303	291
17	T1810	ALTAYYAR	1432	277	945	210	64	T4100	MCDC	2060	470	1336	254
18	T2001	CHEMANOL	2060	402	1239	419	65	T4150	ARDCO	2060	457	1240	363
19	T2010	SABIC	2060	466	1364	230	66	T4160	THIMAR	2060	412	1280	368
20	T2020	SAFCO	2060	446	1365	249	67	T4180	FITAIHI GROUP	2060	364	1363	333
21	T2040	SAUDI CERAMICS	2060	382	1232	446	68	T4190	JARIR	2060	521	1287	252
22	T2050	SAVOLA GROUP	2060	405	1346	309	69	T4200	ALDREES	2060	442	1318	300
23	T2060	TASNEE	2060	452	1231	377	70	T4220	EMAAR EC	2060	419	1249	392
24	T2080	GASCO	2060	473	1268	319	71	T4250	JABAL OMAR	2060	462	1228	370
25	T2090	NGC	2060	325	1292	443	72	T4280	KINGDOM	2060	413	1342	305
26	T2110	SCC	1717	285	1006	426	73	T4290	ALKHALEEJ TRNG	2060	402	1388	270
27	T2120	SAIC	2060	350	1384	326	74	T4300	DAR AL ARKAN	2060	429	1225	406
28	T2150	ZOUJAJ	2060	426	1291	343	75	T4310	KEC	1883	362	1159	362
29	T2180	FIPCO	2060	372	1353	335	76	T5110	SAUDI ELECTRIC.	2060	457	1381	222
30	T2200	APC	2060	380	1269	411	77	T6001	H B	2060	444	1263	353
31	T2210	NAMA CHEMICALS	2055	426	1213	416	78	T6002	HERFY FOODS	2017	344	1486	187
32	T2230	CHEMICAL	2060	477	1275	308	79	T6004	CATERING	1407	278	932	197
33	T2240	ZAMIL INDUST	2060	389	1253	418	80	T6010	NADEC	2060	427	1291	342
34	T2260	SAHARA	2060	491	1160	409	81	T6020	GACO	2060	324	1346	390
35	T2270	SADAFCO	2060	430	1356	274	82	T6040	TADCO	2060	337	1356	367
36	T2280	ALMARAI	2060	467	1404	189	83	T6050	SFICO	2060	325	1323	412
37	T2290	YANSAB	2060	462	1381	217	84	T6070	ALJOUF	2060	364	1447	249
38	T2300	SPM	2048	389	1172	487	85	T7010	STC	2059	543	1180	336
39	T2320	AL-BABTAIN	2060	411	1298	351	86	T7020	ETIHAD ETISALAT	2011	431	1239	341
40	T2330	ADVANCED	2060	521	1245	294	87	T7040	ATHEEB TELECOM	1841	238	1184	419
41	T2340	ALABDULLATIF	2060	378	1300	382	88	T8040	ALLIANZ SF	2060	368	1288	404
42	T2370	MESC	2056	391	1238	427	89	T8080	SABB TAKAFUL	2060	340	1280	440
43	T2380	PETRO RABIGH	2059	436	1183	440	90	T8140	AL-AHLIA	2060	372	1308	380
44	T3001	HCC	1593	284	918	391	91	T8150	ACIG	2026	365	1254	407
45	T3002	NAJRAN CEMENT	1449	247	876	326	92	T8160	AICC	2060	319	1399	342
46	T3003	CITY CEMENT	1345	261	776	308	93	T8180	SAGR INSURANCE	2060	310	1328	422
47	T3010	ACC	2060	410	1195	455	94	T8210	BUPA ARABIA	2060	448	1304	308
48	T3020	YSCC	2060	420	1245	395	95	T8240	ACE	2060	385	1310	365
49	T3030	SCC	2060	451	1261	348	96	T8250	AXA COOPERATIVE	2060	395	1338	327
50	T3040	QACCO	2060	403	1241	416	97	T8260	GULF GENERAL	2013	321	1382	310
.	.	.	.	.	.	.	98	T8270	BURUJ	1988	315	1347	326
.	.	.	.	.	.	.	99	T8300	WATANIYA	1929	333	1272	324
.	.	.	.	.	.	.	100	TTASI	TTASI	2060	539	1141	380

As shown in the flowchart, it starts by loading a given daily prediction file. With each Up prediction a bet is made: a bet is a buy trade of a certain amount of money. A bet is made subject to satisfying some buying rules, such as the availability of funds and the number of open bets allowed. Each bet has a target selling price equal to $(1 + V) * Buy\ price$, and a holding window equal to the lookback window of $w = 20$ trading days. The holding window determines the maturity date of the bet. With each Down prediction, the oldest open bet is closed at the current day's closing price. In the case of both Flat and Down predictions, all open bets are closed when they reach or exceed their target price or maturity date. To make the simulation more realistic, a limit on the number of open bets that can be generated for a given stock is enforced. This is conservatively set to 1 in this study.

Based on the above trading strategy, the investment return was calculated for each stock and compared with the generic baseline investment strategy of buy and hold: where a stock is

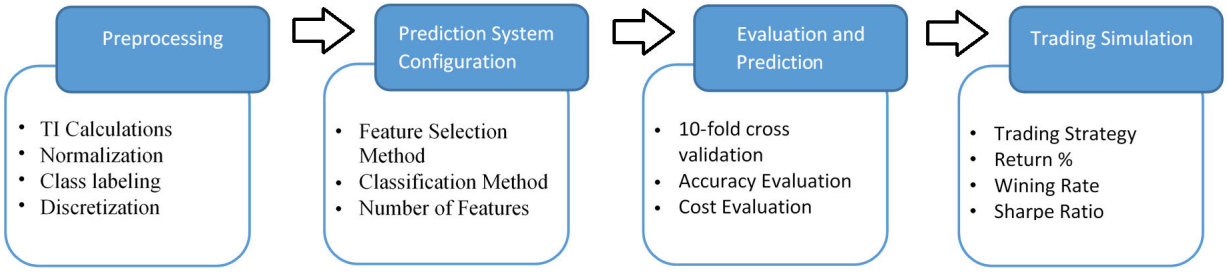

FIGURE 4. Prediction system architecture.

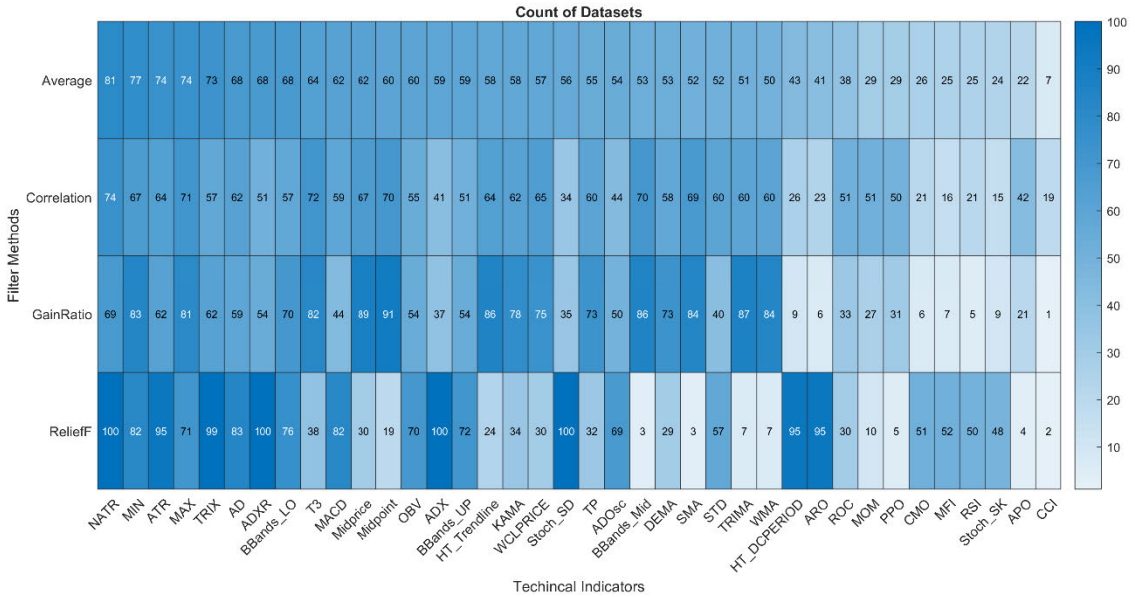

FIGURE 5. Filters heat map showing the count of datasets where a TI is ranked in the top 20. Four filter methods are shown. The top row shows the average, which is used as a guide for sorting TIs.

bought at the beginning of the period and sold at its end. The experiments used a prediction period from July 1, 2015 to March 31, 2018; the first half of 2015 was used for training the classifiers. The classifiers were trained using data from six months to predict the next period of six months, and then the training and forecasting was repeated six months later. This process mitigates the effect of dataset shift or concept shift, which occur when the unseen testing data tends to differ from the training data in the distribution of all or some of its features, or class boundaries [40]. This phenomenon is present in most practical applications [41]. It is certainly evident in the Saudi stock market, because of political and macroeconomic changes during the period under study. In this research, several periods were considered, including a no retraining option, to finally settle on a six-month window, which produced more acceptable results.

In addition to the investment return percentage, the simulation measured the following three investment performance indicators:

- **The number of bets.** This is the total number of bets made on a given stock.
- **Winning rate** (or hit rate). This is the ratio of the number of successful bets to the total number of bets.
- **Sharpe ratio.** This is used in finance to measure the performance of an investment, or trading strategy, after adjusting for risk. This is defined as in (16), where $E(R_A)$ is the return of an asset A, σ_A is the standard deviation of asset A, and R_f is the risk-free rate. The ratio measures the average return achieved above the risk-free rate, per unit of standard deviation σ. A greater standard deviation or volatility is associated with a greater investment risk.

$$Sharpe = \frac{E(R_A) - R_f}{\sigma_A} \tag{16}$$

C. RANKING TIS WITH FILTER METHODS

For each of the 100 datasets, each filter method ranked the 50 TIs differently. This research aimed to find the TIs that were ranked highly by the majority of the datasets. Therefore, for each TI, we counted the number of datasets for which the TI was ranked among the top 20. This procedure was repeated for each filter method. The combined results are shown in Figure 5 as a heat map, in which each row corresponds to a filter method and each column corresponds to a TI. The top row of the heat map shows the average for all filter methods and provides a guide for sorting the TIs.

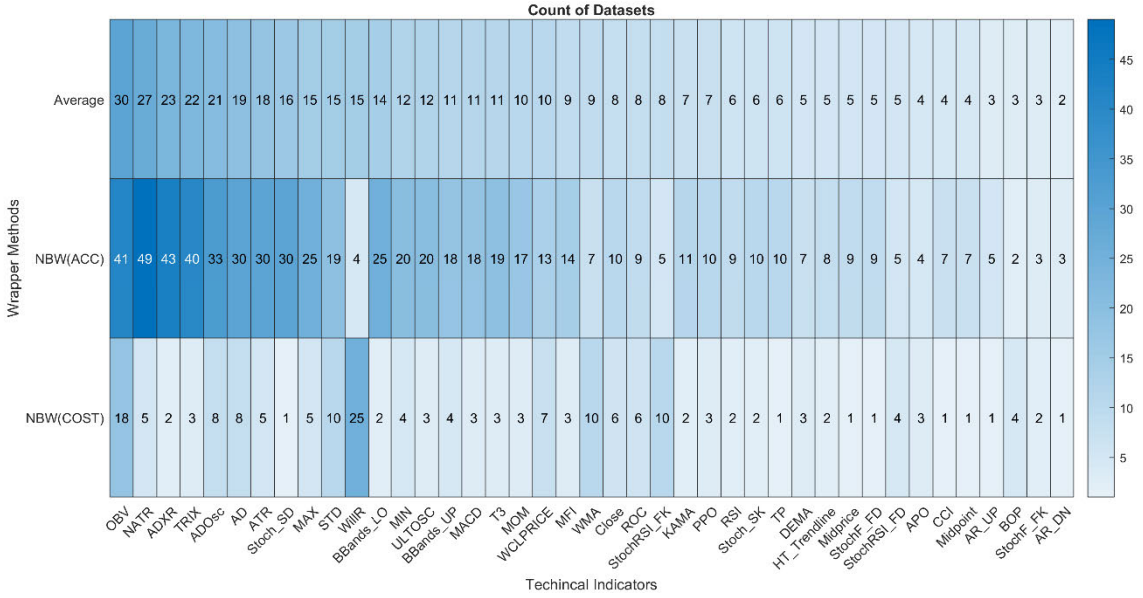

FIGURE 6. Wrappers heat map showing the count of datasets where a TI is a member of their resultant subsets. Two wrapper methods are shown. The top row shows the average, which is used as a guide for sorting TIs.

As shown by the heat map, the NATR TI is ranked in the top 20 by all 100 datasets when applying the ReliefF filter method. It is ranked in the top 20 by 74 datasets with the Correlation method; this is the highest count for this filter. However, with GainRatio, NATR is ranked in the top 20 by only 69 datasets. On average, it is ranked in the top 20 by 81 datasets. Four TIs (NATR, ADXR, ADX, and Stoch_SD) are ranked in the top 20 by all 100 datasets when applying the ReliefF filter method.

The heat map shows which TIs are more significant when applying each filter method. Only the 38 TIs that are ranked highly by all filter methods are shown in the heat map. The remaining TIs are omitted even if they rank highly with one or two filters but not all three.

D. RANKING TIS WITH WRAPPER METHODS

Wrappers generate a subset of most relevant TIs for each dataset, not a ranked list of all 50 TIs, like filters. Therefore, for each TI, we counted the number of datasets in which the TI was a member of the resultant subset of the dataset. Figure 6 shows the results of the NB accuracy wrapper and NB cost wrapper. The top row of the heat map shows the average for the two wrapper methods and provides a guide for sorting the TIs. The first observation is that there is a much smaller number of datasets per TI for both wrappers. This is because of the size of the resultant subsets, which is eight on average for the accuracy wrapper and only two for the cost wrapper. The heat map shows that the NATR TI is the most relevant for the accuracy wrapper because it appeared in 49 subsets, while the WillR TI is the most relevant for the cost wrapper because it appeared in 25 subsets. However, the OBV TI is the most relevant on average. The heat map does not show whole resultant subsets of mixable TIs, as selected by wrappers. It is interesting to see the largest subset of highly ranked TIs for each wrapper, because these wrappers remove redundant features. The whole subset that includes the greatest number of highly ranked TIs for the accuracy wrapper contains 11 TIs {AD, ADXR, ARO, ATR, BBands_UP, MFI, Midpoint, NATR, OBV, STD, T3}, and the best whole subset for the cost wrapper contains five TIs {AD, ADOsc, StochF_FK, StochRSI_FK, WillR}.

E. AGGREGATED RESULTS

This study conducted a total of 180 experiments. Each of the nine classifiers was subjected to 20 different groups of datasets (100 datasets each), generated by the filter and wrapper methods. The filter methods generated 18 groups of datasets (six per method): the six groups contained the top 5, 10, 20, 30, 40, and 50 TIs. The wrapper methods generated one group of subsets per method, making a total of 20 groups of datasets. Each experiment was measured using three metrics: a) accuracy, b) average cost, and c) return percentage. The results were aggregated for each classifier, and shown in Figs. 7 to 11 as boxplots and graphs, to simplify visual representation. The y-axis scale is fixed for all subplots, to enable quick comparison; this caused some boxplot whiskers to be trimmed. Each figure has three rows (one per metric) and five columns (one per feature selection method). The first two rows (a and b) use boxplots to present results, while the third row uses graphs with two y-axes (blue and red) to plot the investment strategy return percentage (as bars) and the Sharpe ratio (as lines). The best performance for a metric is highlighted by a box with a thick border (for the subplot) and an asterisk (for the best performing group of datasets).

Table 9 shows, for each classifier, the average results of the three metrics when applied to each of the 20 groups of datasets. The rightmost column shows the overall average for each classifier, along with a statistical significance mark.

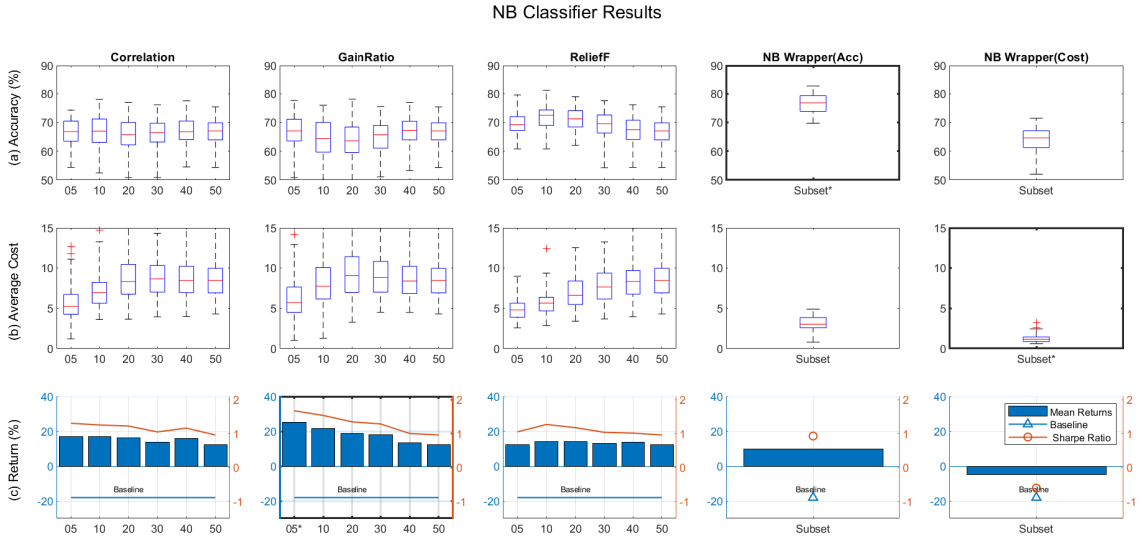

FIGURE 7. Best accuracy for NB classifier is achieved by NB accuracy wrapper. Best cost is achieved by NB cost wrapper. Best Sharpe ratio is achieved by GainRatio filter with five TIs.

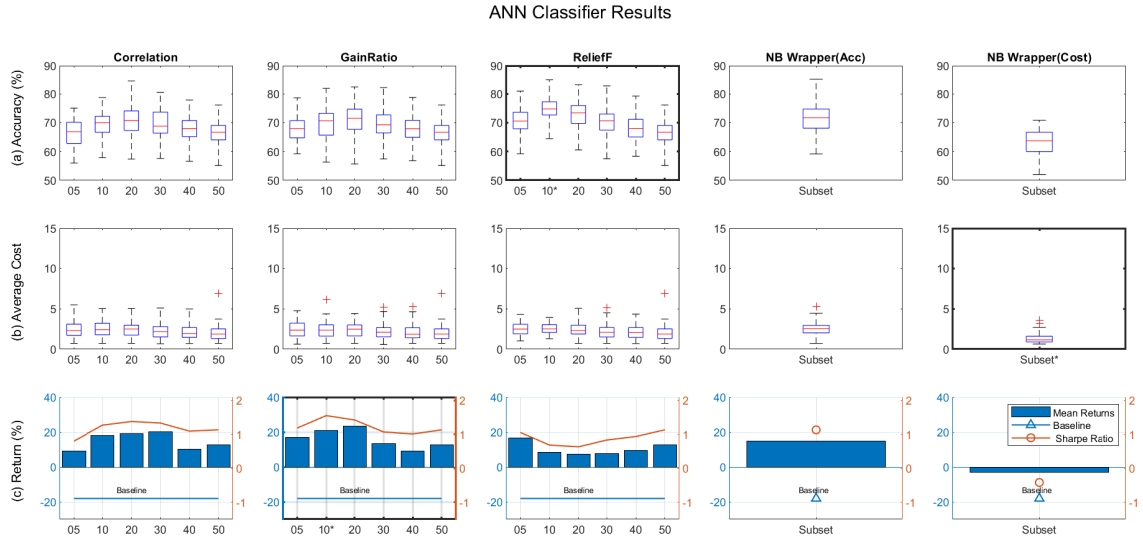

FIGURE 8. Best accuracy for ANN classifier is achieved by ReliefF filter with 10 TIs. Best cost is achieved by NB cost wrapper. Best Sharpe ratio is achieved by GainRatio filter with 10 TIs.

The Wilcoxon signed-rank test [42] was used to compare the distribution of two paired groups: the baseline classifier CSFTNB and each other classifier. A statistical significance mark is attached to each classifier average in the table: "v" indicates that the classifier outperformed the baseline, "$*$" indicates that it underperformed the baseline, and no mark indicates no statistically significant difference. Finally, the Friedman test [43] was used to confirm that there was a statistically significant difference between all classifiers' results for each metric used. Both Wilcoxon and Friedman tests use a significance level of $\alpha = 0.05$.

1) ACCURACY RESULTS

It was expected that the best accuracy score for the NB classifier would be achieved by wrappers, because these wrappers use NB to select TI subsets. Indeed, NB achieved 76.78 ± 3.28 with the accuracy wrapper, which was the highest for NB, making it the third-best classifier, following SVM and FTNB, and this was followed by ANN. The accuracy wrapper produced high accuracy, particularly with ANN, M-NB, and M-ANN, achieving their third-best scores. Table 4 focuses on filter methods only: the classifiers are ordered by their best filter method score. The second-best and third-best scores are listed, to show the effect of the number of features. It can be observed that the best accuracy scores were achieved by most classifiers using 10 and 30 TIs. However, 20 TIs is the mode for all top three scores. Table 4 shows that ReliefF was the best performing filter for accuracy, followed by GainRatio. Generally, from the boxplots and Table 9, the four accuracy classifiers ANN, SVM, NB, and FTNB show similar curves:

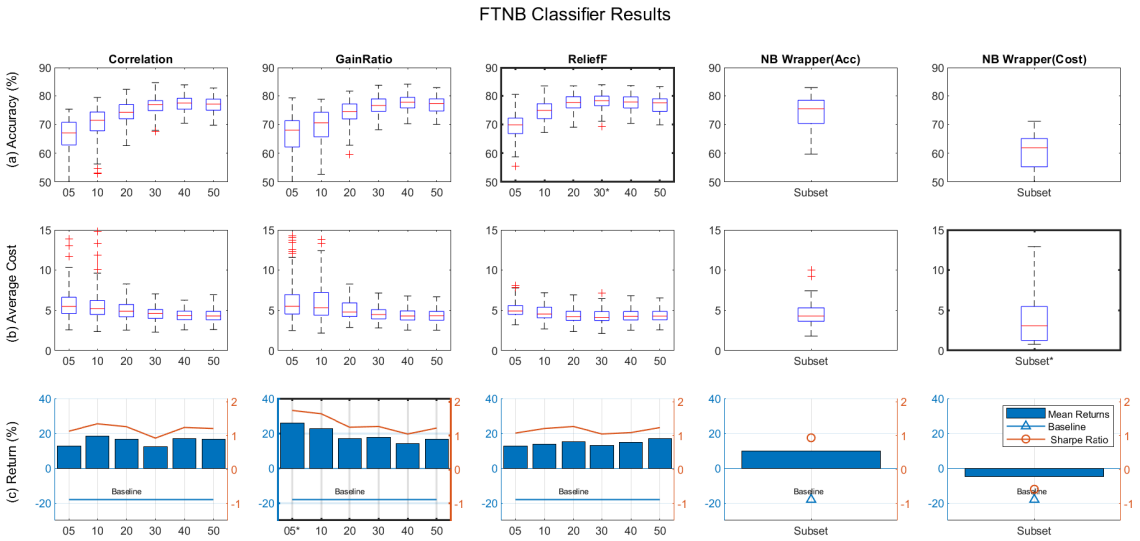

FIGURE 9. Best accuracy for FTNB classifier is achieved by ReliefF filter with 30 TIs. Best cost is achieved by NB cost wrapper. Best Sharpe ratio is achieved by GainRatio filter with five TIs.

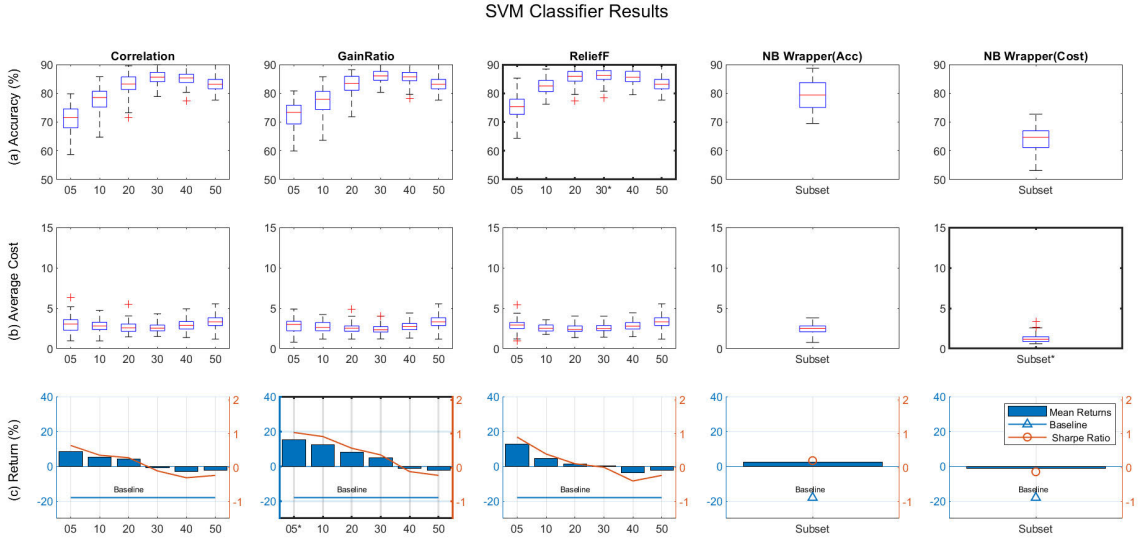

FIGURE 10. Best accuracy for SVM classifier is achieved by ReliefF filter with 30 TIs. Best cost is achieved by NB cost wrapper. Best Sharpe ratio is achieved by GainRatio filter with five TIs.

TABLE 4. Classifiers' best accuracy scores using filter methods.

	1st performing group			2nd performing group			3rd performing group		
Classifier	**Filter**	**# TIs**	**Median**	**Filter**	**# TIs**	**Median**	**Filter**	**# TIs**	**Median**
SVM	ReliefF	30	86.15 ±2.22	GainRatio	30	86.01 ±2.29	ReliefF	20	85.83 ±2.37
FTNB	ReliefF	30	78.06 ±2.86	ReliefF	40	77.64 ±2.89	ReliefF	20	77.58 ±2.90
ANN	ReliefF	10	75.01 ±3.85	ReliefF	20	73.01 ±4.76	GainRatio	20	71.04 ±5.34
M-FTNB	ReliefF	30	73.85 ±3.10	ReliefF	20	73.56 ±3.09	ReliefF	40	73.36 ±3.15
NB	ReliefF	10	71.86 ±4.02	ReliefF	20	71.23 ±3.80	ReliefF	5	69.48 ±4.02
M-NB	ReliefF	10	68.47 ±3.87	ReliefF	20	67.96 ±3.87	GainRatio	5	65.84 ±3.91
M-ANN	ReliefF	10	66.45 ±3.79	ReliefF	20	65.65 ±4.07	GainRatio	20	64.69 ±3.96
CSFTNB	GainRatio	40	64.84 ±4.17	Correlation	50	64.83 ±4.10	GainRatio	50	64.83 ±4.10
M-SVM	Correlation	30	63.56 ±4.25	GainRatio	30	63.54 ±4.41	ReliefF	30	63.54 ±4.37

they peak at either 10 or 30 TIs, with a minimum score at 5 or 50 TIs. Statistically, using the Wilcoxon test, Table 9 shows that CSFTNB's performance was similar to the performance of M-NB and M-ANN, and it outperformed M-SVM, while the other accuracy-based classifiers outperformed it by the accuracy metric, as expected. The accuracy results were also

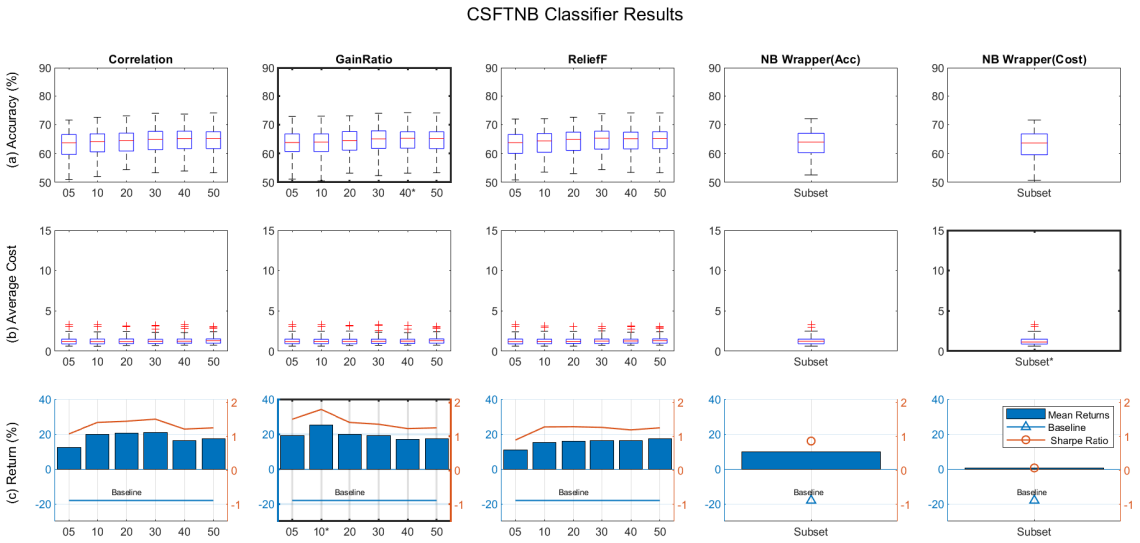

FIGURE 11. Best accuracy for CSFTNB classifier is achieved by GainRatio filter with 40 TIs. Best cost is achieved by NB cost wrapper. Best Sharpe ratio is achieved by GainRatio filter with 10 TIs.

TABLE 5. Classifiers' best cost scores using filter methods.

	1st performing group			2nd performing group			3rd performing group		
Classifier	**Filter**	**# TIs**	**Median**	**Filter**	**# TIs**	**Median**	**Filter**	**# TIs**	**Median**
M-ANN	ReliefF	10	1.14 ±0.32	ReliefF	20	1.17 ±0.35	ReliefF	30	1.20 ±0.40
M-SVM	ReliefF	30	1.22 ±0.40	GainRatio	30	1.22 ±0.40	GainRatio	40	1.22 ±0.40
CSFTNB	GainRatio	10	1.28 ±0.48	ReliefF	5	1.28 ±0.50	ReliefF	10	1.29 ±0.49
M-FTNB	ReliefF	5	1.37 ±0.54	GainRatio	5	1.49 ±0.51	Correlation	5	1.52 ±0.51
M-NB	ReliefF	5	1.70 ±0.70	Correlation	5	2.43 ±1.41	GainRatio	5	2.61 ±1.44
ANN	Correlation	50	2.00 ±0.88	GainRatio	50	2.00 ±0.88	ReliefF	50	2.00 ±0.88
SVM	GainRatio	30	2.43 ±0.56	ReliefF	20	2.44 ±0.48	GainRatio	20	2.53 ±0.57
FTNB	ReliefF	30	4.26 ±0.87	ReliefF	40	4.32 ±0.84	ReliefF	20	4.32 ±0.85
NB	ReliefF	5	4.93 ±1.30	Correlation	5	5.63 ±1.97	ReliefF	10	5.67 ±1.38

TABLE 6. Classifiers' best investment returns.

Classifier	**Filter**	**# TIs**	**Return%**	**STD**	**Sharpe**	**Hit Rate**	**# Bets**
M-SVM	Correlation	50	0.0006	0	-	1	0.03
M-ANN	Correlation	20	0.1497	0.0777	1.93	0.58	3.18
CSFTNB	GainRatio	10	0.2517	0.14	1.8	0.53	28.58
FTNB	GainRatio	5	0.2607	0.1492	1.75	0.53	26.6
NB	GainRatio	5	0.2529	0.1507	1.68	0.53	27.6
ANN	GainRatio	20	0.2345	0.1641	1.43	0.55	9.74
M-FTNB	GainRatio	30	0.2009	0.1415	1.42	0.49	36
M-NB	ReliefF	20	0.1349	0.1107	1.22	0.5	37.63
SVM	GainRatio	5	0.1508	0.1453	1.04	0.5	13.34

tested using the Friedman test: the resulting statistic was 126.320, with p = 1.637e-23, which confirms that the classifiers' accuracy results had different distributions.

2) COST RESULTS

The boxplots show that seven of the nine classifiers achieved their best cost score with the use of the NB Cost wrapper. The remaining two classifiers, M-ANN and M-SVM, achieved their best cost scores (1.14 ± 0.32 and 1.22 ± 0.40, respectively) with the use of ReliefF. These two results are the overall best for all classifiers. As before, Table 5 focuses on filter methods only: classifiers are ordered by their best filter method score. Second-best and third-best scores are listed, to show the effect of the number of features. Most classifiers achieved their best cost scores using 10 and 30 TIs; however, five TIs is the mode for all of the top three scores. As with the accuracy metric, ReliefF, followed by GainRatio, is the best performing filter for cost. Generally, from the boxplots and Table 9, it is clear that three of the five cost-sensitive classifiers (M-ANN, M-SVM, and CSFTNB)

TABLE 7. Classifiers' prediction distribution.

Classifier	**Filter**	**# TIs**	**Up**	**Down**	**Flat**
M-SVM	Correlation	50	0.005%	0.823%	99.173%
M-ANN	Correlation	20	2.5%	11.2%	86.3%
CSFTNB	GainRatio	10	30%	19%	51%
FTNB	GainRatio	5	26%	25%	49%
NB	GainRatio	5	27%	24%	49%
ANN	GainRatio	20	9%	28%	63%
M-FTNB	GainRatio	30	38%	23%	39%
M-NB	ReliefF	20	32%	26%	42%
SVM	GainRatio	5	10%	17%	72%

TABLE 8. Classifiers' best investment returns using filter methods.

	1st performing group				2nd performing group				3rd performing group			
Classifier	**Filter**	**# TIs**	**Return%**	**Sharpe**	**Filter**	**# TIs**	**Return%**	**Sharpe**	**Filter**	**# TIs**	**Return%**	**Sharpe**
M-SVM	Correlation	50	0.00 ±0.00	-	GainRatio	50	0.00 ±0.00	-	ReliefF	40	0.00 ±0.00	-
M-ANN	Correlation	20	0.15 ±0.08	1.93	Correlation	40	0.14 ±0.20	0.67	ReliefF	10	0.13 ±0.14	0.92
CSFTNB	GainRatio	10	0.25 ±0.14	1.8	Correlation	30	0.21 ±0.14	1.51	Correlation	20	0.20 ±0.14	1.45
FTNB	GainRatio	5	0.26 ±0.15	1.75	GainRatio	10	0.23 ±0.14	1.65	Correlation	10	0.19 ±0.14	1.35
NB	GainRatio	5	0.25 ±0.15	1.68	GainRatio	10	0.22 ±0.14	1.54	GainRatio	20	0.19 ±0.14	1.35
ANN	GainRatio	20	0.23 ±0.16	1.43	GainRatio	10	0.21 ±0.13	1.56	Correlation	30	0.20 ±0.15	1.34
M-FTNB	GainRatio	30	0.20 ±0.14	1.42	ReliefF	20	0.17 ±0.13	1.38	Correlation	40	0.16 ±0.13	1.22
M-NB	ReliefF	20	0.13 ±0.11	1.22	ReliefF	30	0.12 ±0.11	1.07	GainRatio	40	0.11 ±0.12	0.94
SVM	GainRatio	5	0.15 ±0.15	1.04	ReliefF	5	0.13 ±0.14	0.9	GainRatio	10	0.12 ±0.13	0.92

TABLE 9. Aggregated average results of accuracy, average cost, and return percentage.

Metric	**Filter**	**Correlation**						**GainRatio**						**ReliefF**						**NB Wrapper (Acc)**	**NB Wrapper (Cost)**	**Average & Significance**
	Ranking	**top05**	**top10**	**top20**	**top30**	**top40**	**top50**	**top05**	**top10**	**top20**	**top30**	**top40**	**top50**	**top05**	**top10**	**top20**	**top30**	**top40**	**top50**	**Subset**	**Subset**	
Accuracy	**CSFTNB**	62.86	63.3	63.89	64.42	64.78	64.83	63.28	63.63	64.25	64.66	64.84	64.83	62.87	63.47	64.32	64.78	64.77	64.83	63.49	63.01	64.05
	NB	66.3	66.47	65.45	66.06	66.9	66.98	66.78	63.84	63.42	65.14	66.92	66.98	69.48	71.86	71.23	69.09	67.39	66.98	76.78	63.97	67.40v
	FTNB	65.94	70.23	74.28	76.57	77.29	76.85	66.27	69.05	74.33	76.52	77.51	76.87	69.88	74.66	77.63	77.96	77.65	76.93	74.4	59.97	73.54v
	ANN	66.8	69.63	70.71	69.47	67.84	66.43	67.96	69.59	71.04	69.33	67.86	66.43	70.51	75.01	73.01	70.52	68.13	66.43	71.5	63.23	69.07v
	SVM	71.08	77.43	83.1	85.58	85.26	83.21	72.7	77.24	83.2	86.01	85.73	83.21	75.42	82.51	85.83	86.15	85.57	83.21	79.32	64.01	80.79v
	M-NB	64.86	65.03	62.36	61.89	62.63	62.86	65.84	63.86	60.25	61.01	62.67	62.86	65.57	68.47	67.96	65.03	63.17	62.86	67.32	63.25	63.99
	M-FTNB	64.24	67.35	70.44	72.22	73.05	72.89	65.78	67.51	69.82	72.04	73.16	72.9	65.13	69.53	73.61	73.91	73.34	72.83	67.97	63.22	0.05v
	M-ANN	63.02	63.93	64.5	64.05	63.53	63.07	63.43	64.29	64.69	64.04	63.53	63.07	64.04	66.45	65.65	64.48	63.71	63.07	64.79	62.85	64.01
	M-SVM	62.98	63.3	63.48	63.56	63.46	63.32	63.04	63.15	63.39	63.54	63.53	63.32	63.04	63.41	63.52	63.54	63.49	63.32	63.32	62.7	**63.32***
Average Cost	**CSFTNB**	1.29	1.29	1.3	1.32	1.33	1.36	1.29	1.28	1.29	1.3	1.33	1.36	1.28	1.29	1.29	1.31	1.34	1.36	1.29	1.26	1.31
	NB	5.63	7.46	8.89	8.91	8.74	8.62	6.24	8.8	9.71	9.38	8.78	8.62	4.93	5.67	6.97	7.98	8.63	8.62	3.05	1.24	**7.34***
	FTNB	5.93	5.69	5	4.56	4.36	4.38	6.33	6.11	5.05	4.58	4.37	4.34	5.1	4.7	4.29	4.27	4.34	4.34	4.51	3.69	**4.80***
	ANN	2.51	2.51	2.42	2.22	2.1	2	2.42	2.39	2.35	2.13	2.02	2	2.54	2.55	2.42	2.17	2.11	2	2.5	1.33	**2.23***
	SVM	3	2.81	2.63	2.56	2.87	3.36	2.85	2.68	2.53	2.43	2.75	3.36	2.88	2.57	2.44	2.56	2.82	3.36	2.42	1.29	**2.71***
	M-NB	2.43	4.71	7.45	7.96	7.84	7.59	2.61	5.37	8.13	8.31	7.8	7.59	1.7	2.94	5.11	6.67	7.57	7.59	1.71	1.26	**5.62***
	M-FTNB	1.52	2.01	2.73	3.16	3.47	3.73	1.48	2.19	2.87	3.13	3.46	3.75	1.36	1.79	2.54	3.03	3.47	3.77	1.43	1.26	**2.61***
	M-ANN	1.25	1.23	1.21	1.23	1.24	1.25	1.24	1.21	1.21	1.22	1.24	1.25	1.22	1.14	1.17	1.2	1.23	1.25	1.19	1.26	1.22v
	M-SVM	1.25	1.24	1.23	1.22	1.22	1.23	1.26	1.24	1.23	1.22	1.22	1.23	1.26	1.23	1.22	1.22	1.22	1.23	1.23	1.27	1.23v
Return Percentage	**CSFTNB**	0.12	0.2	0.2	0.21	0.16	0.17	0.19	0.25	0.2	0.19	0.17	0.17	0.11	0.15	0.16	0.16	0.16	0.17	0.1	0.01	0.16
	NB	0.17	0.17	0.16	0.14	0.16	0.12	0.25	0.22	0.19	0.18	0.13	0.12	0.12	0.14	0.14	0.13	0.14	0.12	0.1	-0.05	**0.14***
	FTNB	0.13	0.19	0.17	0.12	0.17	0.16	0.26	0.23	0.17	0.18	0.14	0.17	0.13	0.14	0.15	0.13	0.15	0.17	0.1	-0.05	**0.15***
	ANN	0.09	0.18	0.19	0.2	0.1	0.13	0.17	0.21	0.23	0.13	0.09	0.13	0.17	0.08	0.07	0.08	0.09	0.13	0.15	-0.03	**0.13***
	SVM	0.09	0.05	0.04	-0.01	-0.03	-0.02	0.15	0.12	0.08	0.05	-0.01	-0.02	0.13	0.04	0.01	0	-0.04	-0.02	0.02	-0.01	**0.03***
	M-NB	0.02	0.06	0.04	0.06	0.11	0.1	0.09	0.1	0.1	0.06	0.11	0.1	0.05	0.06	0.13	0.12	0.09	0.1	0.07	0.01	**0.08***
	M-FTNB	0.02	0.06	0.07	0.12	0.16	0.12	0.1	0.12	0.11	0.2	0.1	0.16	0.11	0.12	0.17	0.13	0.14	0.15	0.05	0.01	**0.11***
	M-ANN	0.02	0.08	0.15	0.07	0.14	0.06	0.02	0.04	0.11	0.11	0.04	0.06	-0.02	0.13	0.08	0.09	0.06	0.06	0.03	0	**0.07***
	M-SVM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00***

- The symbols v and * that appear in the rightmost column denote a statistically significant difference between the classifiers' results and those of the baseline classifier CSFTNB, based on the Wilcoxon signed-rank test.
- v indicates that the classifier outperformed, and * indicates that the classifier underperformed, the baseline classifier.

maintain low-cost scores with the use of more TIs; however, M-NB and M-FTNB scores are negatively affected by increasing the number of TIs, with their worst results for 30 or more TIs.

Statistically, using the Wilcoxon test, CSFTNB outperformed all other classifiers except M-ANN and M-SVM, in respect of the cost metric. The Friedman test was also performed on the cost results: the statistic was 130.147, with $p = 2.638e\text{-}24$, which confirms that the classifiers' cost results had different distributions.

3) TRADING SYSTEM SIMULATION RESULTS

The baseline results of the buy-and-hold strategy for the period was an aggregated negative return of -0.18 ± 0.93,

with a Sharpe ratio of −0.19. Figure 1 shows the market index performance for the period, which explains the negative results. Table 6 shows the maximum returns achieved by each classifier. All classifiers were able to achieve a positive return as their best result. The table also shows the hit rate and the average number of bets made per stock, for each trading system configuration. All results in the table were calculated per stock, and then aggregated and averaged over the 99 stocks. In the table, the classifiers are sorted by their Sharpe ratios; in this case, the Sharpe ratio is simply the return divided by the standard deviation.

M-SVM is top-ranked in the table because its Sharpe ratio is affected by division by the zero standard deviation. M-SVM made only three bets overall, with a 100% hit rate. This extremely risk-averse classifier is followed by M-ANN, which returned 15% on investments of three bets (on average) per stock, with a Sharpe ratio of 1.93. The best overall performance was achieved by the CSFTNB classifier, with a 1.8 Sharpe ratio. CSFTNB made over 28 bets (on average) per stock, with a winning rate of 53%. Good results were also achieved by the accuracy classifiers FTNB and NB. The CSFTNB returns were statistically greater than those of all other classifiers. The Friedman test statistic was 102.747, with $p = 1.171e\text{-}18$, which confirms that the classifiers' results had different distributions.

To help explain the performance of the classifiers, Table 7 shows the prediction distribution of each classifier with the same configuration as in Table 4. The class distribution for all datasets combined was 15%, 24%, and 61% for Up, Down, and Flat, respectively. Table 7 shows that SVM and ANN were very strongly biased toward the majority class, with 72% and 63% respectively. The cost-sensitive versions of these two classifiers enforced this bias, rather than mitigating it. However, the cost-sensitive versions of NB and FTNB did reduce the majority class bias. The extreme case was observed with M-SVM, for which over 99% of predictions were in the Flat class. This explains why M-SVM achieved the best results for the cost metric and the worst for both the accuracy and investment return metrics.

Finally, it is important to note that, in contrast with the accuracy and cost metrics, GainRatio was the best performing filter for investment return. The best scores were achieved by most classifiers using five or 20 TIs; however, 10 TIs was the mode for all of the top three scores, as shown in Table 8.

VI. CONCLUSION

This work contributed a detailed investigation of the selection of a minimal number of relevant TIs to forecast the direction of movement of stock prices. The results show that choosing a large number (more than 30) of TIs was likely to reduce prediction accuracy, increase misclassification cost, and reduce investment return. More specifically, results show that the highest-accuracy scores were produced with at least 10 TIs, the lowest-cost scores with five TIs, and the best investment performance with five or 10 TIs. The research found that classifiers that achieved the highest-accuracy scores and those that achieved the lowest-cost scores did not produce the best investment performance when used to simulate a realistic trading strategy. CSFTNB, a cost-sensitive fine-tuning of the NB classifier, achieved the highest balance between investment return and associated risk.

The proposed CSFTNB classifier augments the classical NB classifier with a fine-tuning phase, in which the value probabilities are modified or fine-tuned to reduce the classification cost. Unlike the MetaCost method [27], no relabeling is made to the datasets; thus, the instance classes remain unchanged. However, the changes and adjustments are made to the probability values used by NB in such a manner that it is more likely to produce the class with the lowest prediction cost, and not the correct class. Stock market trend prediction is an interesting problem that can be tackled in the context of cost-sensitive learning because the cost of misclassification, or risk, is a significant factor that traders need to consider.

Several feature selection configurations and various alternatives for the number of TIs were tested, to measure and compare the performance of the nine prediction models. The performance of each setup was measured using three metrics: classification accuracy, misclassification cost, and the expected investment return of a trading system simulation. Data from the Saudi stock market was used to construct and test the prediction models.

This research also presented a ranking of 50 widely used TIs, using three different feature selection methods, as well as two wrapper methods: accuracy-based and cost-based. The ranking heat maps can serve as a guide for selecting TIs.

As future work, we intend to investigate different feature selection techniques, to select from a broader set of indicators, including (in addition to TIs) fundamental and sentiment indicators that best reduce the cost of misclassification. We also aim to incorporate cost-sensitive learning with other fine-tuning methods, such as [32], [44], as well as studying the effect of noise and developing techniques—such as those reported in [45]—to mitigate the impact of noise.

APPENDIX

The following five figures from Figure 7 to Figure 11 show detailed results for each classifier.

REFERENCES

[1] F. S. Board, *Artificial Intelligence and Machine Learning in Financial Services*. Accessed: Jan. 30, 2018. [Online]. Available: http//www.fsb.org/2017/11/artificialintelligence-and-machine-learning-in-financialservice/

[2] Y. Shynkevich, T. M. McGinnity, S. A. Coleman, A. Belatreche, and Y. Li, "Forecasting price movements using technical indicators: Investigating the impact of varying input window length," *Neurocomputing*, vol. 264, pp. 71–88, Nov. 2017.

[3] Y. Kara, M. A. Boyacioglu, and Ö. K. Baykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5311–5319, May 2011.

[4] K. J. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, nos. 1–2, pp. 307–319, Sep. 2003.

[5] M. Qiu and Y. Song, "Predicting the direction of stock market index movement using an optimized artificial neural network model," *PLoS ONE*, vol. 11, no. 5, May 2016, Art. no. e0155133.

[6] G. S. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques—Part II: Soft computing methods," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5932–5941, Apr. 2009.
[7] M. P. Naeini, H. Taremian, and H. B. Hashemi, "Stock market value prediction using neural networks," in *Proc. Int. Conf. Comput. Inf. Syst. Ind. Manage. Appl. (CISIM)*, Oct. 2010, pp. 132–136.
[8] S. O. Olatunji, M. Al-Ahmadi, M. Elshafei, and Y. Fallatah, "Forecasting the saudi arabia stock prices based on artificial neural networks model," *Int. J. Intell. Inf. Syst.*, vol. 2, no. 5, pp. 77–86, 2013.
[9] R. Alkhareif, "Are there significant premiums in the Saudi stock market," *Financ. Res. Lett.*, vol. 18, pp. 108–115, Aug. 2016.
[10] Y.-W. Si and J. Yin, "OBST-based segmentation approach to financial time series," *Eng. Appl. Artif. Intell.*, vol. 26, no. 10, pp. 2581–2596, Nov. 2013.
[11] R. C. Cavalcante, R. C. Brasileiro, V. L. F. Souza, J. P. Nobrega, and A. L. I. Oliveira, "Computational intelligence and financial markets: A survey and future directions," *Expert. Syst. Appl.*, vol. 55, pp. 194–211, Aug. 2016.
[12] A. Gupta and D. S. D. Sharma, "A Survey on Stock Market Prediction Using Various Algorithms," *Int. J. Comput. Technol. Appl.*, vol. 5, no. 2, pp. 530–533, Apr. 2014.
[13] C. Zopounidis, E. Galariotis, M. Doumpos, S. Sarri, and K. Andriosopoulos, "Multiple criteria decision aiding for finance: An updated bibliographic survey," *Eur. J. Oper. Res.*, vol. 247, no. 2, pp. 339–348, Dec. 2015.
[14] L. Guo, F. Shi, and J. Tu, "Textual analysis and machine leaning: Crack unstructured data in finance and accounting," *J. Financ. Data Sci.*, vol. 2, no. 3, pp. 153–170, Sep. 2016.
[15] W.-C. Hong, M.-W. Li, J. Geng, and Y. Zhang, "Novel chaotic bat algorithm for forecasting complex motion of floating platforms," *Appl. Math. Model.*, vol. 72, pp. 425–443, Aug. 2019.
[16] Y. Dong, Z. Zhang, and W.-C. Hong, "A hybrid seasonal mechanism with a chaotic cuckoo search algorithm with a support vector regression model for electric load forecasting," *Energies*, vol. 11, no. 4, p. 1009, 2018.
[17] G.-F. Fan, L.-L. Peng, and W.-C. Hong, "Short term load forecasting based on phase space reconstruction algorithm and bi-square kernel regression model," *Appl. Energy*, vol. 224, pp. 13–33, Aug. 2018.
[18] B. M. Henrique, V. A. Sobreiro, and H. Kimura, "Stock price prediction using support vector regression on daily and up to the minute prices," *J. Finance Data Sci.*, vol. 4, no. 3, pp. 183–201, 2018.
[19] M.-Y. Chen and T.-H. Chen, "Modeling public mood and emotion: Blog and news sentiment and socio-economic phenomena," *Future Gener. Comput. Syst.*, vol. 96, pp. 692–699, Jul. 2019.
[20] P. Ou and H. Wang, "Prediction of stock market index movement by ten data mining techniques," *Mod. Appl. Sci.*, vol. 3, no. 12, pp. 28–42, Dec. 2009.
[21] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, "Evaluating multiple classifiers for stock price direction prediction," *Expert Syst. Appl.*, vol. 42, no. 20, pp. 7046–7056, Nov. 2015.
[22] Z.-H. Zhou and X.-Y. Liu, "On multi-class cost-sensitive learning," *Comput. Intell.*, vol. 26, no. 3, pp. 232–257, 2010.
[23] C. X. Ling and V. S. Sheng, "Cost-sensitive learning and the class imbalance problem," in *Encyclopedia of Machine Learning*. Berlin, Germany: Springer, 2011, pp. 231–235.
[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
[25] L. Jiang, C. Qiu, and C. Li, "A novel minority cloning technique for cost-sensitive learning," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 29, no. 4, 2015, Art. no. 1551004.
[26] L. Jiang, C. Li, and S. Wang, "Cost-sensitive Bayesian network classifiers," *Pattern Recognit. Lett.*, vol. 45, pp. 211–216, Aug. 2014.
[27] P. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," in *Proc. 5th Int. Conf. Knowl. Discovery Data Mining*, Aug. 1999, pp. 155–164.
[28] V. S. Sheng and C. X. Ling, "Thresholding for making classifiers cost-sensitive," in *Proc. AAAI*, Jul. 2006, pp. 476–481.
[29] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[30] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275–306, 2010.
[31] K. El Hindi, "Fine tuning the Naïve Bayesian learning algorithm," *J. AI Commun.*, vol. 27, no. 2, pp. 133–141, Apr. 2014.
[32] A. Alhussan and K. El Hindi, "Selectively fine-tuning Bayesian network learning algorithm," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 30, no. 8, 2016, Art. no. 1651005.
[33] A. Grimes, *The Art and Science of Technical Analysis: Market Structure, Price Action, and Trading Strategies*, vol. 546. Hoboken, NJ, USA: Wiley, 2012.
[34] R. Bellman and Karreman Mathematics Research Collection, *Adaptive Control Processes: A Guided Tour* (Princeton Legacy Library). Princeton, NJ, USA: Princeton Univ. Press, 1961. [Online]. Available: https://books.google.com.sa/books?id=POAmAAAAMAAJ
[35] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2015, pp. 1200–1205.
[36] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.
[37] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using support vector machines," *Inf. Sci.*, vol. 286, pp. 228–246, Dec. 2014.
[38] R. R. Bouckaert, "Weka manual for version 3-6-0," Univ. Waikato, Hamilton, New Zealand, Tech. Rep., 2008.
[39] TA-Lib. *Technical Analysis Library—Home*. Accessed: May 18, 2019. [Online]. Available: https://ta-lib.org/
[40] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," in *Pattern Recognition*. Amsterdam, The Netherlands: Elsevier, 2012. doi: 10.1016/j.patcog.2011.06.019.
[41] N. Adams, "Dataset shift in machine learning," *J. Roy. Stat. Soc. A, Statist. Soc.*, vol. 173, no. 1, p. 274, 2010. doi: 10.1111/j.1467-985X.2009.00624_10.x.
[42] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
[43] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, 1940.
[44] D. M. Diab and K. M. El Hindi, "Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification," *Appl. Soft Comput.*, vol. 54, pp. 183–199, May 2017.
[45] K. El Hindi, "A noise tolerant fine tuning algorithm for the Naïve Bayesian learning algorithm," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 26, no. 2, pp. 237–246, Jul. 2014.

YAZEED ALSUBAIE received the M.Sc. in computer science from the University of Colorado, USA. He is currently pursuing the Ph.D. degree with the Computer Science Department, King Saud University, Riyadh, Saudi Arabia. He has several years of experience in IT and telecommunication fields in technical and managerial positions. His research interests include machine learning and computational finance.

KHALIL EL HINDI received the B.Sc. degree from Yarmouk University and the M.Sc. and Ph.D. degrees from the University of Exeter, U.K. He is currently a Professor with the Department of Computer Science, King Saud University. His research interests include machine leaning and data mining. He is particularly interested in improving the classification accuracy of Bayesian classifiers and developing new similarity metrics for instance-based learning.

HUSSAIN ALSALMAN received the B.Sc. and M.Sc. degrees in computer science from King Saud University (KSU), Riyadh, Saudi Arabia, and the Ph.D. degree in artificial intelligence from U.K. He worked for several years as a Consultant for a number of companies in private sector and institutes in government sector, Saudi Arabia. From 2009 to 2014, he chaired Computer Science Department, College of Computer and Information Sciences, KSU. He is currently a Staff Member with the Computer Science Department, KSU. He was a member of review board of *Saudi Computer Journal*, from 2004 to 2014.