Capstone Project

Jian (Kevin) Wang

Machine Learning Engineer Nanodegree

February 24, 2017

Predicting long-term stock gain using machine learning methods

Definition

Project Overview

This capstone project was inspired by my personal interest and passion in long term investing in the stock market, as well as Udacity's suggested problem on Investment and Trading Investment for the capstone project. The goal of this project is to build a classification model that predicts whether or not the stock price will rise at least 10% in one year for stocks in the consumer staples sector. Unlike other approaches focusing only on the stock historic pricing data for prediction of future price, herein I utilized both the stock historic pricing and fundamental data for predicting the long-term (one year) stock price trend.

With stock market index such as the S&P 500 index reaching new highs, the S&P 500 is expected to increase only 4% by the end of this year [1]. John Bogle, the founder of mutual fund giant Vanguard, recently predicted that US stocks will produce only 4% annualized return in the next decade [2]. Recently, Milosevic reported a machine learning based approach using various algorithms to evaluate whether or not a stock's price will increase by 10% in one year [3]. Dai and Zhang used various machine-learning models to forecast the price trend of a single stock using various fundamental data of the stock [4]. In addition, Madge and Bhatt reported using Support Vector Machines to predict the price direction of technology stocks [5].

As an individual investor passionate about learning stock market investing, I believe certain individual stocks may produce superior returns than the others based on stock's fundamental data. Given the abovementioned prediction of US stock market return, a 10% gain on individual stocks will be a superior investment return. In addition, long term gain (one year or more) will result in more favorable tax rates for individual investors. This motivated the work for this capstone project: predicting whether or not a stock will gain at least 10% in one year. For the capstone project, I limited the study of stocks to the consumer staples sector that I believe will have similar characteristics and stock price movements. I am also personally interested in the sector of consumer staples given the relatively less complexity of the business model, slow and steady growth, and good diversification advantage with its relatively low correlation with the overall stock market [6]. The stock fundamental data are available from the SEC's SEC's EDGAR database [9] and stock's historic pricing data are available from Yahoo Finance [11]. The target variable to predict can be determined by whether or not the stock's price rises at least 10% in one year.

Problem Statement

I used supervised learning methods to predict whether or not a stock will rise 10% or more in one year, given the historical stock data, including EPS, P/E ratio, Dividend, Adjusted Closing Price, Market Cap, and Current Ratio (Current Assets / Current Liability), for stocks in the consumer staples sector. For each of the stocks, quarterly financial data from company 10-Q reports as well as the stock pricing data from the end of each quarters were collected and used to predict a binary class variable: 1 if the stock will rise at least 10% in one year and 0 otherwise. All input data are of numerical type and the target variable to predict is binary. The data set will be collected from 2009 until 2015 such that the target variable based on stock prices will be known for both the training and test sets. The performance of the classification model was evaluated by overall accuracy and other metric as discussed in the later section. The classification model was built using six supervised learning methods, including Decision Tree, Naïve Bayes, K Nearest Neighbor (KNN) Random Forest and Support Vector Machines (SVM).

The strategy to solve this problem is outlined in these overall steps:

1. Data acquisition: Acquire stock fundamental and historic pricing data
2. Data preprocessing: Explore the data and cleanse redundant and undesirable data. Transform the data by using scaling and normalization techniques.
3. Create training and testing set by shuffling and splitting the data.
4. Build classification models and evaluate model performance.
5. Identify the most important features for classification.
6. Evaluate performance using only the most important features.

The performance of the classification models were compared against a naïve predictor, which always predict the stock price will rise at least 10% in one year (target variable = 1 for all cases), using overall accuracy and other relevant metric. It was anticipated certain classification models would clearly outperform the naïve predictor based on the defined metrics.

Metrics

There are two general metrics to be considered when evaluating a supervised learning model: (1). Model accuracy for measuring how accurately the model makes predictions (2) Model efficiency for measuring how fast the model can be trained and can make predictions. As shown in the following sections, the final data set used for this project includes a total of 575 samples. Each model will be measured in terms of the its efficiency but more emphasis was placed on the model accuracy metric, as the difference in time each model needed for training and prediction is not necessarily substantial given the data set size. The specific metric for measuring model accuracy is described below.

The performance of each supervised learning model will be measured by the overall prediction accuracy and F-beta score (with beta=0.5) to account for both precision and recall on the validation set and the test set.

Accuracy = $\frac{Correctly\ predicted\ cases}{Total\ number\ of\ cases}$

F-beta score = $(1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$

With beta = 0.5
(The formula for F-beta score is cited from the Udacity's "Finding Donors for CharityML" project).

precision = tp / (tp + fp) where tp is the number of true positives and fp is the number of false positives [13]

recall = tp / (tp + fn) where tp is the number of true positives and fn is the number of false negatives [14]

For example, consider the following table showing the number of cases according to the actual class label and the predicted class label.

		Predicted Class	
		1	0
Actual Class	1	5	2
	0	3	6

In the above example, tp = 5, fp=3, tn = 6, fn=2. precision, recall, F-beta and Accuracy score can then be calculated accordingly based on the above-defined formula.

As shown in the following section, the final dataset for building and testing the classification model contains over 65% of records where the stock gained at least 10% in one year. So the dataset is skewed towards more positive samples (with target variable = 1). Given this unbalance, the overall accuracy score is not as effective as the F-beta score (with beta=0.5) for evaluating the model performance. Furthermore, I believe for investment return, it is more important to be able to make correct prediction than covering correct prediction on more positive cases (all the stocks that will gain at least 10% in one year). In other words, the precision is more important than the recall value. Therefore, both the overall accuracy score and F-beta score were included in the metrics but the F-beta score (with beta=0.5) was placed with greater emphasis.

Analysis

Data Exploration 1 – Data acquisition

There is no input dataset directly available for this project. I acquired the following data from publicly available domains as described below.

(1) List of S&P 500 stocks in the consumer staples sector

I downloaded the list of stocks in the S&P 500 index from the consumer staples sector (37 stocks), based on the Global Industry Classification Standard [7] from Wikipedia [8].

(2) Stock's historic quarterly financial data

For S&P 500 stocks from the consumer staples sector, I extracted the stock historic quarterly financial data (2009-2015), including report date, Revenues, EPS (Earnings Per Share), Dividend, Current Assets and Current Liabilities, from the 10-Q filings from the SEC's EDGAR database [9] using the pystock-crawler python package [10]. Extracted financial data after 2015 were removed.

(3) Stock historic pricing data

For each of those historic financial data records at quarter ending dates, I extracted the Adjusted Close Price from Yahoo Finance [11] using the yahoo-finance Python API: [12]. If the stock price information was not available at the quarter ending date from the 10-Q filing, the Adjusted Close price at the next trading date when the price data is available was extracted. Similarly, I extracted the Adjusted Close Price for each stock one year (365 days) from each of the quarter ending dates. In addition, use this Python API to extract the current price and market capitalization data from Yahoo Finance.

All quarterly financial data for the following four stocks (KFT, HANS, WFMI, WXZ) were excluded from the input file for running the Python code, as attempts to extract historic price data from Yahoo Finance using the Python API for each of these four stocks was unsuccessful for some or all quarter ending dates.

In addition to extracting the historic price and market capitalization data from Yahoo Finance, the following calculations were made to create new columns of data that were added to the input file for classification:

- p_e ratio: p_e = Adjusted Close Price / (eps_diluted * 4)
- Estimated market capitalization (in billions) at the quarter ending date = (Current Market Cap * Adjusted Close Price at quarter ending date) / Current stock price)

- Current ratio = Current Assets / Current Liability
- Gain is the target variable to be predicted. Gain = 1 if Adjusted Close price is at least 10% higher in one year for each stock at each of the quarter ending dates. Gain = 0 otherwise.

Data Exploration 2 – Data examination

After acquiring the data, the following steps describe the further examination of the data.

Table 1 shows a sample of the initial dataset to be used for classification. And Table 2 shows a summary of the initial dataset that was acquired. All seven input data variables for classification are of numerical type with disparate ranges. All predicting variables have positive values except eps_diluted, with a minimal value being negative, and dividend, with a minimal value being 0. "gain", the target for prediction is a binary variable: 1 if the stock gained at least 10% in one year and 0 otherwise. About 65% of records have a positive target value (target variable (gain_ = 1), as shown in Table 3. No missing value was present in this acquired initial dataset.

Upon further examination of the dataset, the following abnormalities were identified:

(1) Duplicate records

There were certain duplicate records with the same key (defined as the stock symbol concatenated with the report date) in the data set. Table 4 shows an example of the duplicate records. Duplicate records were subsequently removed.

(2) Eight of 587 data records had unique stock fundamentals, i.e. negative earnings per share, as shown in Table 5. For the purpose of this project to build a classifier for stock long-term gain, these records with negative earnings will be deleted, as the number of these records is relatively small and in reality it is much more challenging to predict the return of stocks with negative earnings and therefore these records with negative earnings per share were subsequently removed.

Exploratory Visualization

To visualize the data and explore the overall data distribution (after removing redundant records and records with negative earnings per share), I plotted a histogram for each of the predicting variables. Shown in Figure 1 is the histogram plot for the market capitalization variable (market_cap), in billions. Figure 2 shows the histogram plot for the P/E ratio variable (p_e).

As exemplified by the histogram plots for these two predicting variables, the distribution of the data is skewed towards one end of the X-axis. In addition, the predicting variables appear to be based on different units and therefore different scales.

Algorithms and Techniques

The problem dealt with in this project is a binary classification problem. The input data set contains seven predicting variables, all of numerical type. The target variable for prediction contains two classes: 1 and 0. This classification problem is readily addressable by various machine-learning algorithms for classification.

Previously, Milosevic utilized multiple machine learning algorithms, including Decision trees, Support Vector Machines (SVM), Random Forest, Logistic regression, Naïve Bayes, etc., to predict long term stock price change, with the Random Forest algorithm achieving the most effective result [3]. Madge and Bhatt used Support Vector Machines to predict the stock price direction, with definite predictive capability for long-term prediction [5]. With a goal of predicting long-term stock price gain similar to these previous studies but using a unique combination of formulating the classification problem (predicting whether or not a stock price will rise at least 10% in one year) with a specific set of stocks (consumer staples), I will take a similar approach to utilize six machine learning algorithms for classification for the problem in this project. The six algorithms include: (1) Decision Tree (2) Logistic Regression (3) Naïve Bayes (4) K Nearest Neighbors (KNN) (5) Support Vector Machines (SVM) (6) Random Forest.

Each of these classification models utilize specific algorithm to take the feature values as input and produce a class label for a test case. Below are the descriptions on how each of these algorithms works.

For the Decision Tree classification algorithm, it is aimed at creating a tree like model that can predict the class of a target variable based on the input variables. Shown in Figure 3 (a) is an example Decision Tree classification model for predicting whether a person survived or died. Each node in the tree represents an input variable. Based on possible values of the input variable, there are different paths leading to children from the node. A leaf in the decision tree represents a value for the target variable. Essentially, the values of the input variables will lead to a path from the root node to the leaf, where a classification label is determined based. Such a decision tree can be "learned" based on a training set. Basically, the source data can be split into subsets of data based on testing a specific attribute value. Such a process is repeated on every resulting subset of data in a recursive manner, leading to the construction of a decision tree for classifying new samples [16].

Strengths of the model include: (1) It can generate understanable rules (2) It can handle both categorical and continuous variables (3) It can provide a good indication on which features are important for prediction. (4) Not much computation is required. The decision trees model performs well when both categorical and continuous variables are present in the data set and that much computation is not desired for performing classification. Weaknesses of the model include: (1) It does not perform well for predicting continous variable. (2) The error rate will be large when Decision Trees are applied where there are a lot of classes and the number of training samples is relatively small. Decision Trees do not perform well for predicting continuous variables or when the number of classes is large and the number training samples is relatively small [17]. These discussions on strengths and weaknesses are cited from the work I did from Udacity Project "Finding Donors".

The Logistic Regression algorithm for classification assigns a weight for each predicting variable that determines how each feature can contribute to the final class of the target variable. Each feature's weight is multiplied with the feature's value for every predicting variable. The resulting products for all the features are summed up and subject to a logistic function $\sigma(t) = \frac{1}{1+e^{-t}}$. The output of this logistic function represents a probability value between 0 and 1. This probability value will then be used to determine whether or not target variable belongs to one class vs the other (with 1 being very certain and 0 being unlikely and 0.5 being a usual threshold), in the case of binary classification [18, 19]. Figure 3 (b) shows a Logistic Regression model.

Strengths of Logistic Regression include: (1) It is easy to implement and relatively straightforward [20]. (2) It can output a probability value with which a specific threshold can be applied to decide the class label suite the specific business needs [21] (3) It is robust to small noise [21]. Logistic Regression performs well when there might be small noise in the data and a probability value is needed to be associated with the class label. Weaknesses of Logistic Regression include: (1) If the number of predicting attributes is large, the Logistic Regression model might not be optimal without first reducing the predicting features by pre-processing [20] (2) If there is dependence of some data points with others, Logistic Regression could over-weigh the importance of those observations and resulting in bias. In these situations, Logistic Regression could perform poorly [22]. Some of these discussions about Logistic Regression are cited from the work I did from Udacity Project "Finding Donors".

The Naïve Bayes classifier represents a family of simple probabilistic classification models that apply Baye's theorem, describing the probability an event happens based on prior knowledge of certain conditions possibly related to the event [23], that strongly assumes that features are independent. The Naïve Bayes model for classification assigns class labels to cases represented by vectors of feature values, with a finite set of values for the class labels. The Naïve Bayes classifier assumes the value of a feature is not dependent on the value of any other feature [24]. One of the advantages of Naïve Bayes classifier is that it can be trained on a small number of cases in order to estimate the parameters for classification [24]. An obvious disadvantage is that Naïve Bayes classifier will not work well when there is dependence among the features.

For the K Nearest Neighbors (KNN) algorithm for classification, a sample is classified by a majority vote of its K nearest neighbors, i.e. the class label that is most common among the K nearest neighbors. Figure 3 (c) shows an example of K Nearest Neighbor classification The distance can be measured by a variety of metrics, including the Euclidean distance. [25]. Advantages of the KNN algorithm includes: (1) Zero learning cost (2)No assumptions need to be made about the classification concepts. (3)Simple procedures can be utilized for learning complex processes with local approximation [26]. Disadvantages of KNN include: (1) It is not possible to interpret the model. (2) For large dataset, it is computationally expensive to identify the K nearest neighbors [26].

The Support Vector Machines (SVM) algorithm for classification constructs a hyperplane or set of hyperplanes to separate and classify samples. The hyperplane is set up such that the nearest training data points have the largest distance to the hyperplane in any class [27]. Figure 3 (d) shows an example of Support Vector Machines for classification.

Strengths of SVMs include: (1) The model can be applied on data sets with high dimensions (2) Relatively easy to train with no local optimal. SVMs can perform well when there is a lot of dimensions in the data set [28]. Weaknesses of SVMs include: (1) Limited in the choice of kernels [29] (2) The performance of the classification model may be extremely slow in test phase [29] (3) It is not quite transparent of the results from the SVMs as SVMs can not represent a metric based on a simple parametric function of all the data [30].

A Random Forest classifier constructs multiple decision trees during training and output the class label that occurs most often among the individual decision trees [31]. A Random Forest classification model grows a forest of many decision trees, with each tree grown on an independent set of cases that were sampled from the training set at random with replacement. At each node: m variables are selected at random out of all possible variables and the best split on the selected m variables is identified. These decision trees are grown to maximum depth and the class label for a test case is determined by a vote from all the trees [32].

Advantages of the Random Forest classifier include: (1) It runs efficiently on large datasets. (2). It can provide an estimate on the importance of features for its predictive capability (3). It can handle unbalanced datasets. [33]. (4) It is highly accurate in many data sets [34]. Disadvantage of the Random Forest classifier includes that it could overfit for certain datasets with noisy classification tasks [35].

These six algorithms were chosen as some of these algorithms have been successfully applied for stock price movement prediction in previous studies as cited above. In addition, these algorithms are established algorithms that have been successfully utilized for classification in many other prediction problems in the realm of machine learning. Finally, these algorithms can be readily implemented using the scikit-learn package (http://scikit-learn.org). Given the unbalanced nature of this data set (more samples with target variable being positive) and that later on it will be useful to evaluate the importance of features, the Random Forest classifier may be a good candidate for this project. Actually, Random Forest has been reported previously to have achieved the best performance for stock price movement prediction [3]. In addition, the Support Vector Machines algorithm has been applied on other previous studies for stock price prediction [4,5]. For this capstone project, all six algorithms will be tested and evaluated and the best classification model will be identified and applied on the dataset.

I plan to apply logarithm to rescale the predicting features data and subsequently normalize each feature such that each feature will have 0 mean and unit standard deviation. Then I plan to shuffle the dataset and split the dataset to a training set and a testing set. I will use cross-validation to search through a number of combinations of parameters in order to identify the optimal parameters for the classifier to perform the best from cross-validation in the training set based on the evaluation metrics. I will then re-run the classifier with the optimized parameters on the testing set.

For the Support Vector Machines (SVM) algorithm, I will test the prediction by using both the linear and rbf kernels, aiming to build a classifier that is not over-fitting the data while not under-fitting. I will search through a number of different values for the Cost parameter in order to identify the optimal parameter combination based on cross validation.

For the K Nearest Neighbors classifier, I will search through various combinations of the n_neighbors parameter, algorithm parameter and the weights parameter in order to identify the optimal parameter combination based on cross validation.

For the Random Forest classifier, I will search through various combinations of the n_estimators parameter, the criterion parameter, and the max_features parameter in order to identify the optimal parameter combination based on cross validation.

Finally, I plan to assess the importance of the features for predicting the long term stock price gain and subsequently re-test the classification on the testing set using the most important features.

Benchmark

A naïve predictor will be utilized as the main benchmark model. Such a naïve predictor will predict every stock at any given quarter ending date to rise at least 10% in one year. I believe this is a good benchmark model as the goal of this project is to develop a more accurate prediction model for investment decision-making. In addition, performance and results from other studies [3,4,5] will be discussed and compared with this study, even though the approaches and datasets from those studies are not necessarily the same as in this project.

Methodology

A Python script was written to implement this project and presented in a Python Jupyter Notebook. The sklearn package and other Python API packages [10, 12, 15] were also used.

Data preprocessing

Based on the abovementioned data exploration processes, the following preprocessing steps were performed on the initial dataset:

1. Remove duplicate records

As described in the previous section, duplicate records were removed. Each record in the resulting data set after removing duplicates has a unique key, the combination of stock symbol and report date.

2. Remove records where earnings per share is negative.

As discussed in the previous section, eight of 587 data records had negative earnings per share, shown in Table 5. As justified in the previous section, the number of these records is small and these records in a practical setting may not be of great value for long-term stock price movement prediction. Therefore, these eight records were removed.

3. Data transformation

As discussed in the previous data exploration section, each predicting variable has a skewed distribution towards one end of the X-axis and these predicting variables appear to be on different scales. In order to make the predicting variables on similar scales with a less skewed distribution, the following steps were performed to transform the data.

(A) The values of the predicting variables were applied by logarithm (after adding a very small number: 0.001, as some initial values were 0).
(B) Subsequently, each predicting variable was normalized such that each variable across the records has 0 mean and unit standard deviation.

Figure 4 shows the distribution of market capitalization (market_cap, in billions) in the transformed dataset. Figure 5 shows the distribution of P/E ratio (p_e) in the transformed dataset. Table 6 shows a summary of the transformed dataset. Table 7 shows the distribution of records based on target variable (gain) for preprocessed dataset. As exemplified on these figures and tables, the predicting variables were on a more comparable scale and the distribution is closer to normal after transformation.

Implementation

All implementations for this capstone project were carried out in Python, using various libraries, including scikit-learn.

For the evaluation metrics measuring the model prediction accuracy, I utilized the fbeta_score function and the accuracy_score function in sklearn.metrics to calculate the F-beta score and accuracy score, respectively. For evaluating the Naïve predictor, I also calculated the Accuracy and F-beta score using the following formulas:

Accuracy = $\frac{Correctly\ predicted\ cases}{Total\ number\ of\ cases}$

F-beta score = $(1+\beta^2)\cdot\frac{precision\cdot recall}{(\beta^2\cdot precision)+recall}$

with beta = 0.5
(The formula for F-beta score is cited from the Udacity's "Finding Donors for CharityML" project).

precision = tp / (tp + fp) where tp is the number of true positives and fp the number of false positives [13]

recall = tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives [14]

For calculating the time each model took to train the model and to make predictions, I utilized the time module in Python to calculate the elapsed time.

For the input dataset after transformation, I randomly shuffled the data set and split the dataset to a training (80% of data) and a testing set (20% of data), using the train_test_split function from sklearn.cross_validation. A random seed was set in order to be able to reproduce the results.

For the six classification algorithms, Decision Tree, Naïve Bayes, K Nearest Neighbors, Support Vector Machines, Logistic Regression and Random Forest, I used the default parameters for all algorithms except K Nearest Neighbors (where I had to set n_neighbors=4 instead of the default 5) to train each of the classification models using the 1%, 10% and 100% of the training set and then make predictions in the testing set. A random seed was set wherever needed when initializing each model in order to be able to reproduce the results.

Overall, the implementation process was fairly straightforward without complications. Utilizing part of the Python code in a previous project (Finding Donors) has been helpful to make the process smooth.

Refinement

After training each model using the initial parameters and making predictions on the testing set, I selected top three algorithms with the best performance of F-beta score in the testing set. Subsequently for each of these top algorithms I used six fold cross validation to adjust the model parameters and derive a set of parameters that produced the best performance for each of the three models. Six fold cross validation was chosen such that each fold (~76 cases (~13% of all data): the total number of training cases (460)/6) in the cross validation contains reasonably sufficient number of cases for testing the model performance for the purpose of fine-tuning.

The F-beta score for the six classification models in the testing set using the initial parameters ranged from ~0.60 to ~0.75 and the accuracy score ranged from ~0.55 to ~0.71. The K Nearest Neighbors, Support Vector Machines and Random Forest were the top three performers based on the F-beta score and were subsequently used for further parameter optimization using cross validation, as described in the following.

For the K Nearest Neighbors (KNN) classifier, I searched through various combinations of the n_neighbors parameter, algorithm parameter and the weights parameter in order to identify the optimal parameter combination based on cross validation. For the Support Vector Machines (SVM) algorithm, I tested the prediction by using both the linear and rbf kernels, aiming to build a classifier that is not over-fitting the data while not under-fitting. I searched through a number of different values for the Cost parameter in order to identify the optimal parameter combination based on cross validation. For the Random Forest (RF) classifier, I searched through various combinations of the n_estimators parameter, the criterion parameter, and the max_features parameter in order to identify the optimal parameter combination based on cross validation.

Table 8 shows the performance of these three top classification models before and after optimization, as compared with the benchmark predictor. Both the SVM and KNN Models achieved improved performance while the RF model performance was not further improved after the optimization process. Overall, the RF model achieved the highest performance in terms of both the F-beta score and the Accuracy score.

Finally, the top-performing model, Random Forest, was utilized to assess the importance of features. The feature_importances_ parameter was readily extracted from the optimized Random Forest model. The five most predictive features were identified and reported. These five most predictive features were subsequently utilized to train the Random Forest classifier. The classifier was evaluated when using only the five most important features as compared with the model trained on full data (results shown in Table 9).

Results

Model Evaluation and Validation

The final derived model is a Random Forest classifier using all features in the input data. Random Forest, K Nearest Neighbors and Support Vector Machines were the top three performers, measured by the F-beta score and Accuracy score, as shown in Figure 6(a) and 5(b). Note that the performance generally improved when more training data were utilized to build the classification model, which makes sense. Also, the classifier performance was better in the training data than in the testing data in general. But the performance in the testing set has much higher importance to evaluate the classifier in order to circumvent model over-fitting. The time each model needed to be trained and to make predictions was also measured and shown in Figure 6(a) and 5(b) but due to the relatively small sample size, the time metric was not emphasized for this project but should be considered when the dataset size is large.

The top three performers, Random Forest, K Nearest Neighbors and Support Vector Machines were subsequently subject to further model refinement by using a six fold cross validation approach tuning a range of parameter combinations for each classification model as described. As shown in Table 8, the performance of the Support Vector Machines and K Nearest Neighbors was improved after model optimization while the performance of the Random Forest classifier did not improve after optimization. Overall, the Random Forest classifier achieved the highest performance, as measured by the Accuracy score and the F-beta score (0.7130 and 0.7597, respectively). I believe these performance metrics were substantive as compared with the performance by the benchmark, a Naïve Predictor that achieved the Accuracy score and the F-beta score of 0.6504 and 0.6993, respectively.

Furthermore, feature importance was evaluated by using the best performing classifier: Random Forest, which provides a useful function to rank the predictive importance of each feature. Using the Random Forest classification model, the importance of features was ranked and the top five important features and their weights are shown in Figure 7. These five features are: market capitalization, revenues, dividend, adjust close price and current ratio, which all make sense as these fundamental data are closely related to a company’s financial performance and therefore its stock price movement. A Random Forest classification model was trained using only these five most predictive features and the performance of the model was compared with the Random Forest model using full data. As shown in Table 9, the performance of the Random Forest classifier decreased when using only the five most predictive features instead of the full data, suggesting that the remaining features still possessed predictive capability for predicting the stock long-term gain. This is consistent with the results in Figure 7, which shows that the cumulative weight of five most predictive features was around 75% with the remaining features containing 25% of weight in terms of making correct predictions in the testing set.

Finally, in order to validate the robustness of the Random Forest classifier, I performed sensitivity analysis for the Random Forest classifier to examine how the classifier's predictive capability was affected when the training and testing set were perturbed, I re-shuffled the dataset using a different random_seed and created a re-shuffled training and testing set for another 19 times. Then I re-evaluated the performance of the Random Forest classifier. Different random_state parameters (1,2,3,...20) were used to run the following code to generate the various performance results for each shuffled testing dataset. The performance metrics for the various shuffled testing set are shown in Table 10. The performance of the Random Forest classifier fluctuated with the differently shuffled datasets with the average F-beta score and Accuracy score from these tests being 0.7411 and 0.6574, respectively.

The results from the model evaluation and validation appear to be overall consistent and reliably reproduced. The results can be trusted for the final justification to be discussed in the following section.

Justification

The final model derived from this project is a Random Forest classifier. Overall, I believe that the Random Forest classifier performed well as the model trained on the training set made reasonably satisfying predictions on a set of unseen testing set, as compared with the benchmark model when measured by the F-beta score and Accuracy score. On the 20 shuffled unseen testing sets, the Random Forest model achieved an average F-beta score of 0.7411. I performed a One-sample t test to test if the average F-beta score from these shuffled testing sets is statistically significant from the F-beta score of the benchmark model (the Naïve predictor). The F-beta score of the benchmark model was 0.6993. My One-sample t test, performed using the scipy.stats Python package [15], showed a p value of 8.997e-05, far below 0.05, a typical threshold for significance testing. This result basically tells that the chance that the mean of observed F-beta scores from the Random Forest classifier from the various testing sets being equivalent to the benchmark F-beta score of 0.6993 is extremely low, indicating the observed F-beta scores from the Random Forest classifier is statistically significant, supporting the validity of the Random Forest classification model for long-term stock price gain prediction.

Performance by the Random Forest classifier from this capstone project for long-term stock price gain prediction is on par with related studies by others for stock price movement prediction. In the study by Milosevic [3], a Random Forest classifier achieved a F-score of 0.751, slightly above the F-beta score reported in this project, for predicting long-term stock price movement using data from different stocks than used in this project. In the study by Dai & Zhang [4], a Support Vector Machines based classification model achieved 79% of predicting accuracy for predicting the stock price trend in 44 days for a single stock. In another study by Madge and Bhatt [5], a Support Vector Machines based classifier could achieve 61.5% of prediction accuracy for the price movement of technology stocks in 90 days. Due to the nature of the data used in this capstone project and the preference of using F-beta score to evaluate the performance metric, I believe the achievement of an average F-beta score of 0.7411 with the Random Forest classifier is on par with other's results.

In summary, I believe the performance achieved by the final model, a Random Forest classifier, in this capstone project is stronger than the benchmark result and on par with the performance result by others in related study. I believe the final solution of using Random Forest for predicting the long-term stock price gain is significant enough to have solved the problem for this capstone project.

Conclusion

Free-Form Visualization

With the prediction of a modest return of the overall stock market [1,2], it is without saying it is critically important to be able to identify individual stocks that can beat the overall stock market index in the long term if one wants to achieve above-market returns. This capstone project aimed to address this problem by using machine-learning methods to build a classification model that can predict whether or not the stock price will rise at least 10% in one year. Six different classification algorithms, Decision Tree, Naïve Bayes, Logistic Regression, K Nearest Neighbors, Support Vector Machines and Random Forest were tested. I found that the Random Forest classifier achieved the best performance on the stock data used in this capstone project. By using a Random Forest classification model based on the stock fundamental and historic pricing data, such a model showed promise in make predictions that are statistically superior than a Naïve predictor, which predicts every stock will rise at least 10% in one year.

I believe the final model, Random Forest classifier, has accomplished the goal set for this project. As shown in Figure 8, the F-beta scores from the 20 shuffled testing sets are shown in a box plot, with the F-beta score from the benchmark model included. The F-beta score from the benchmark model, 0.6993, is obviously outside of the confidence interval of the F-beta scores from the Random Forest classifier. And it shows that the Random Forest classifier achieved clearly higher F-beta score than the benchmark.

However, it should be taken as a grain of salt that making predictions on the stock market return is a very hard problem and there are many variables involved. In this capstone project, only seven features about the stock were utilized for building the classification model and only 7 years (2009-2015) of quarterly financial data for consumer staples stocks were utilized

Reflection

In this capstone project, I tackled a problem that is of personal interest: to build a classification model that can predict whether or not a stock will rise at least 10% in one year. I started out by first defining the problem to be solved and the data sets that are available to support building such a prediction system.

I defined the problem to be a binary classification problem: target variable = 1 if a stock price rises at least 10% in one year and 0 otherwise. I then identified two types of data that can be used as features for making the prediction: stock fundamental data and stock historic price data. Both types of data are publicly available and I utilized existing Python packages that can automatically extract those data.

I searched the literature and found similar stock prediction problems have been tackled by others. The dataset and approach that taken by this capstone project are unique in that only stocks in the consumer staples sector were analyzed and predicted and the time horizon for stock price movement is one year with a threshold of gaining at least 10%. I believe the settings for the prediction system and the prediction models chosen are appropriate.

In the end, I was able to tested six different machine-learning algorithms and identified a model that achieved the best performance.

There are two aspects of the project that I found particularly interesting.

First, the majority of my time spent on this project has been on two areas:

1. Formulating the problem. There are many problems to solve for stock price prediction. For example, one may want to predict the actual stock price in a future date, which can be a regression problem in machine learning. One may also want to predict whether the stock price will go up or down in a future date. Based on my personal interest and passion in long term investing, I decided to formulate the problem for this capstone project to be a classification problem that predicts whether or not a stock will rise at least 10% in price in one year.

2. Acquiring and preprocessing the dataset needed for the project. To complete this capstone project, I spent considerable time on acquiring data from publicly available sources, using Python based APIs. I also spent a large amount of time exploring and transforming the data before the data is ready for the classification problem.

I expect this is typical for any machine learning task: to define the business problem and to prepare the data set will consume a lot of time involved in a machine learning project.

Secondly, it is critically important to select and define a specific benchmark and performance metrics to evaluate the machine-learning model. In this project, I utilized a Naïve predictor model and relied on F-beta score and Overall accuracy score as the performance metrics. I would image that for a different problem, different benchmark or metrics may be needed. For example, if the problem were to predict a future stock price, mean squared error might be better suited for evaluating the performance of a regression model. Again, a reasonable benchmark and sensible performance metrics need to be carefully selected and defined.

Improvement

One potential improvement for this capstone project is to add more features for predicting the target variable. In this project, I selected seven features, including the stock fundamental data and historic pricing data, that were used to predict whether or not the stock price will rise at least 10% in one year. By adding more features, one may be able to further improve the performance of the classification model. In addition, by adding more features, the classification model can identify the specific features that are of the most predictive values. For this capstone project, it turned out that the Random Forest classifier achieved the best performance when using all seven features, consistent with my original reasoning of the selection of features that I believed important for stock price prediction. However, adding additional features may further improve the model performance.

Disclaimer

The methodologies, approaches, opinions and any information presented in this project report are exclusively for the purpose of completing the Capstone Project for the Udacity Machine Learning Nanodegree. Any information presented in this report must not be regarded as trading or investing advice. Trade in the stock market at your own risk.

References:

1. http://www.marketwatch.com/story/sp-500-expected-to-rise-only-4-by-end-of-2017-2016-12-24

2. http://www.marketwatch.com/story/john-bogle-says-you-wont-make-much-money-from-stocks-2015-11-05

3. Milosevic. Equity forecast: Predicting long-term stock price movement using machine learning. 2016. https://arxiv.org/pdf/1603.00751.pdf

4. Dai & Zhang. Machine Learning in Stock Price Trend Forecasting. http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf

5. Madge and Bhatt. Predicting Stock Price Direction using Support Vector Machines. 2015. https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf

6. Schmidt. A Guide To Investing In Consumer Staples. http://www.investopedia.com/articles/economics/08/consumer-staples.asp

7. Global Industry Classification Standard (GICS) https://www.msci.com/gics

8. https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

9. SEC's EDGAR database https://www.sec.gov/edgar/searchedgar/companysearch.html

10. pystock-crawler 0.8.2 https://pypi.python.org/pypi/pystock-crawler

11. Yahoo Finance https://finance.yahoo.com

12. https://pypi.python.org/pypi/yahoo-finance/1.1.4

13.http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

14.http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

15. https://docs.scipy.org

16. https://en.wikipedia.org/wiki/Decision_tree_learning

17. Andrew W. Moore https://pdfs.semanticscholar.org/4049/1dc18f32292d56508729e4d66738999a1542.pdf

18. https://www.quora.com/What-is-logistic-regression.

19. https://en.wikipedia.org/wiki/Logistic_regression

20. Stephen G. Powell, Kenneth R. Baker, Management Science. Chapter 6: Classification and Prediction Methods. PowerPoint slides from: faculty.tuck.dartmouth.edu/images/uploads/faculty/management-science/Ch06.ppt

21. Lalit Sachan 2015, http://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2/

22. http://classroom.synonym.com/disadvantages-logistic-regression-8574447.html

23. https://en.wikipedia.org/wiki/Bayes%27_theorem

24. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

25. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

26. http://www.cs.upc.edu/~bejar/apren/docum/trans/03d-algind-knn-eng.pdf

27. https://en.wikipedia.org/wiki/Support_vector_machine

28. Rutgers University CS 536: Machine Learning Littman (Wu, TA)
http://www.cs.rutgers.edu/~mlittman/courses/ml04/svm.pdf

29. http://www.svms.org/disadvantages.html

30. Laura Auria, Rouslan A. Moro, Berlin, August 2008, Support Vector Machines (SVM) as a Technique for Solvency Analysis
https://core.ac.uk/download/pdf/6302770.pdf

31. https://en.wikipedia.org/wiki/Random_forest

32. http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf

33. http://amateurdatascientist.blogspot.com/2012/01/random-forest-algorithm.html

34. Caruana, Rich; Karampatziakis, Nikos; Yessenalina, Ainur (2008). "An empirical evaluation of supervised learning in high dimensions".Proceedings of the 25th International Conference on Machine Learning (ICML).

35. Segal, Mark R. (April 14 2004). Machine Learning Benchmarks and Random Forest Regression. Center for Bioinformatics & Molecular Biostatistics.

Table 1 Sample of the initial dataset for classification. Note that the symbol and end_date columns are included in this table for illustration purpose and were not used for classification.

symbol	end_date	revenues	eps_diluted	dividend	adj_close	market_cap	current_ratio	p_e	gain
TAP	2009-06-28	7.989000e+08	1.01	0.0000	35.477692	7.626079	1.213746	8.781607	0
CVS	2009-06-30	2.487110e+10	0.60	0.0763	28.506514	30.247070	1.313997	11.877714	0
CL	2009-06-30	3.745000e+09	1.07	0.0000	29.484818	26.172913	1.301710	6.888976	1

Table 2. Summary of the initial dataset for classification

	revenues	eps_diluted	dividend	adj_close	market_cap	current_ratio	p_e	gain
count	5.870000e+02	587.000000	587.000000	587.000000	587.000000	587.000000	587.000000	587.000000
mean	1.082493e+10	0.832845	0.409080	46.301053	38.512371	1.398893	16.231184	0.652470
std	1.952030e+10	0.736859	1.033678	24.146309	47.272328	0.737733	24.771381	0.476592
min	-1.000000e+06	-2.230000	0.000000	6.614944	3.202275	0.567011	-93.809167	0.000000
25%	1.545058e+09	0.555000	0.170000	28.365897	10.111996	0.943127	9.527540	0.000000
50%	3.856000e+09	0.770000	0.320000	42.085853	16.927508	1.180152	13.412638	1.000000
75%	1.268000e+10	1.060000	0.490000	62.361115	44.307588	1.683445	18.597077	1.000000
max	1.193360e+11	14.850000	24.000000	144.295689	227.409768	7.973498	384.444458	1.000000

Table 3. Distribution of records based on target variable (gain) for the initial dataset

	Number of records	**% of records**
gain = 0	204	34.75%
gain = 1	383	65.25%
Total	587	100%

Table 4. An example of duplicate records in the initial dataset for prediction

symbol	end_date	revenues	eps_diluted	dividend	adj_close	market_cap	current_ratio	p_e	gain
DPS	2011-09-30	1.529000e+09	0.71	0.32	33.428684	6.122288	0.982039	11.770663	1
DPS	2011-09-30	1.529000e+09	0.71	0.32	33.428684	6.122288	0.982039	11.770663	1

Table 5. Eight of the 587 data records with negative earnings per share (eps_diluted) that were subsequently removed

symbol	end_date	revenues	eps_diluted	dividend	adj_close	market_cap	current_ratio	p_e	gain
KR	2009-11-07	1.766900e+10	-1.35	0.095	10.493115	9.843211	0.922342	-1.943169	0
COTY	2014-03-31	1.008700e+09	-0.66	0.000	14.420025	10.769862	1.790361	-5.462131	1
TAP	2014-09-30	1.168000e+09	-0.19	0.370	71.294967	15.325617	0.691618	-93.809167	1
CAG	2015-02-22	3.876700e+09	-2.23	0.250	25.654857	11.168159	1.055530	-2.876105	1
KHC	2015-06-28	-1.000000e+06	-0.91	0.000	69.805839	84.953094	1.729740	-19.177428	1
KMB	2015-06-30	4.643000e+09	-0.83	0.880	101.296012	36.090090	0.815669	-30.510847	1
HSY	2015-07-05	1.578825e+09	-0.89	0.486	85.608726	18.223066	0.994637	-24.047395	1
KHC	2015-09-27	6.120000e+09	-0.27	0.550	66.954189	81.483270	1.811869	-61.994619	1

Table 6. Summary of the features in the dataset after transformation

	revenues	eps_diluted	dividend	adj_close	market_cap	current_ratio	p_e
count	5.750000e+02	5.750000e+02	5.750000e+02	5.750000e+02	5.750000e+02	5.750000e+02	5.750000e+02
mean	-4.220778e-16	-5.502845e-17	-4.238156e-17	1.650853e-17	6.893037e-17	-1.351576e-17	1.199041e-16
std	1.000871e+00	1.000871e+00	1.000871e+00	1.000871e+00	1.000871e+00	1.000871e+00	1.000871e+00
min	-2.153817e+00	-6.509534e+00	-2.777110e+00	-3.171120e+00	-1.903354e+00	-1.987851e+00	-4.519135e+00
25%	-8.391889e-01	-4.801034e-01	-3.657461e-02	-6.000808e-01	-7.658231e-01	-7.350420e-01	-5.418344e-01
50%	-1.293127e-01	6.809701e-02	2.905029e-01	9.265700e-02	-2.550157e-01	-1.873314e-01	1.516958e-02
75%	8.081908e-01	6.595075e-01	5.142249e-01	7.635882e-01	6.988949e-01	6.758223e-01	5.625824e-01
max	2.562619e+00	5.468213e+00	2.583680e+00	2.272808e+00	2.319066e+00	4.485686e+00	5.634191e+00

Table 7. Distribution of records based on target variable (gain) for preprocessed dataset

	Number of records	**% of records**
gain = 0	201	34.96%
gain = 1	374	65.04%
Total	575	100%

Table 8. Performance of top three classification models before and after optimization, as compared with the benchmark predictor. RF: Random Forest; KNN: K Nearest Neighbors; SVM: Support Vector Machines.

Metric	**Benchmark Predictor**	**Unoptimized RF Model**	**Optimized RF Model**	**Unoptimized KNN Model**	**Optimized KNN Model**	**Unoptimized SVM Model**	**Optimized SVM Model**
Accuracy Score	0.6504	0.7130	0.7130	0.6261	0.6522	0.6435	0.6609
F-beta score	0.6993	0.7597	0.7597	0.7026	0.7123	0.6846	0.7042

Table 9. Performance of the Random Forest classifier on testing data using the five most predictive features compared with using the full dataset

	Using 5 most predictive features	Using full data
Accuracy Score	0.6609	0.7130
F-beta score	0.7249	0.7597

Table 10. Performance of the Random Forest classifier on differently shuffled testing data using. Different random_state parameters were used to generate differently shuffled training (80% of data) and testing sets (20% of data). The training set was used to train the Random Forest classifier model using default parameters and the testing set was used to evaluate the model performance by Accuracy score and F-beta score.

	Accuracy score	F-beta score
Random_state = 1	0.6435	0.7459
Random_state = 2	0.6435	0.7143
Random_state = 3	0.7130	0.7597
Random_state = 4	0.6435	0.7413
Random_state = 5	0.5565	0.6295
Random_state = 6	0.6261	0.6997
Random_state = 7	0.6783	0.7507
Random_state = 8	0.7043	0.7701
Random_state = 9	0.6870	0.7722
Random_state = 10	0.6087	0.7232
Random_state = 11	0.7391	0.8065
Random_state = 12	0.6783	0.7382
Random_state = 13	0.5739	0.7205
Random_state = 14	0.6870	0.7385
Random_state = 15	0.6957	0.7767
Random_state = 16	0.6696	0.7843
Random_state = 17	0.6261	0.7434
Random_state = 18	0.6783	0.7345
Random_state = 19	0.6348	0.7662
Random_state = 20	0.6609	0.7069
Average	**0.6574**	**0.7411**

Figure 1. Distribution of market capitalization (market_cap, in billions) in the dataset

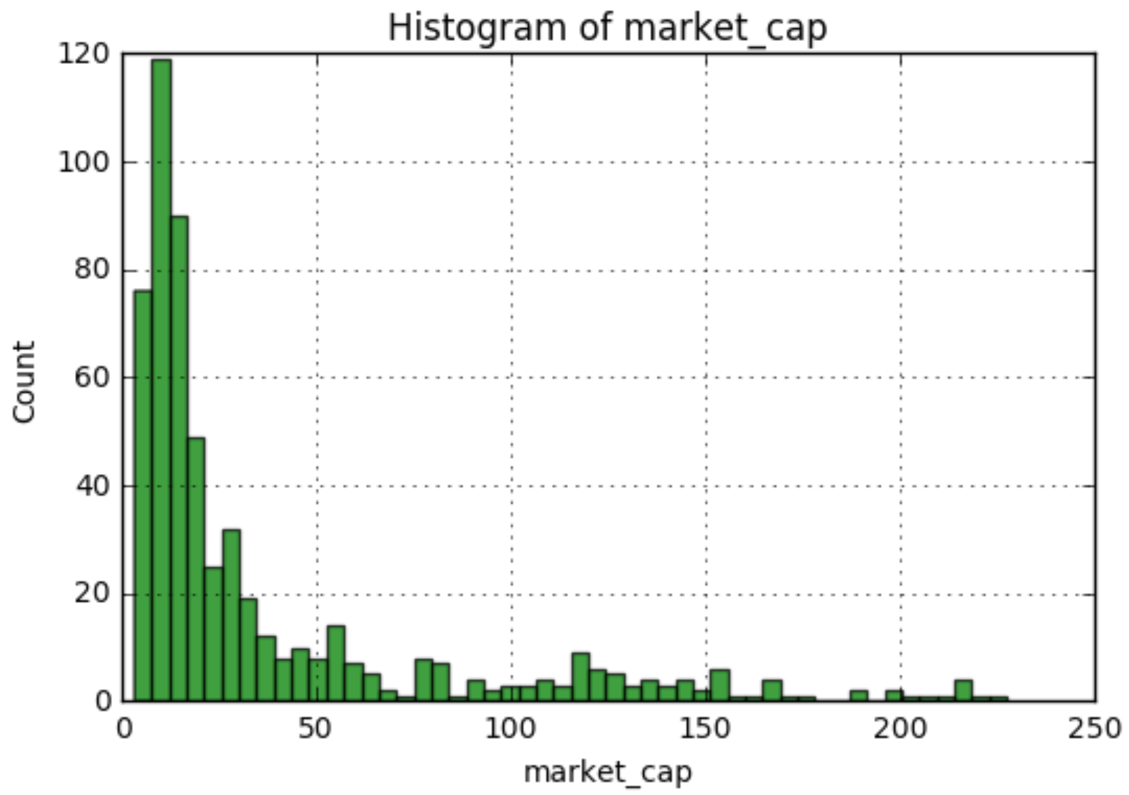


Figure 2. Distribution of P/E ratio (p_e) in the dataset

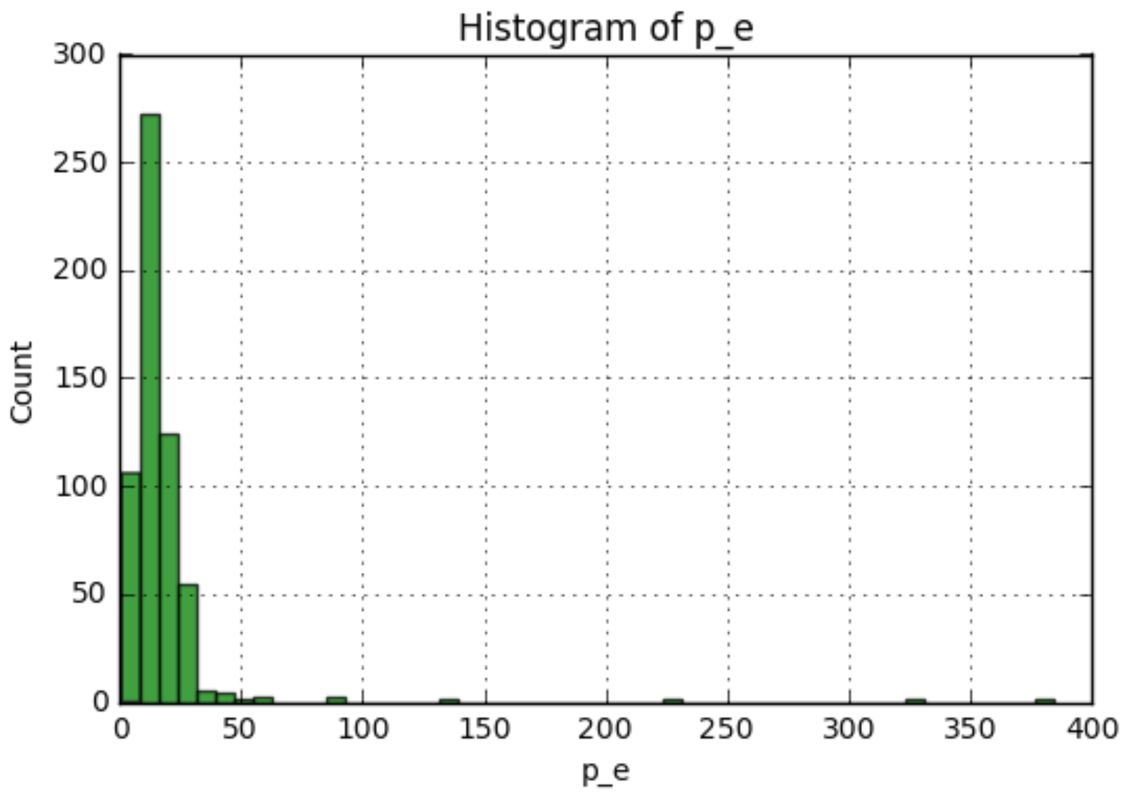


Figure 3 (a). An example Decision Tree classification model for predicting whether a person survived or died. Source: https://upload.wikimedia.org/wikipedia/commons/f/f3/CART_tree_titanic_survivors.png

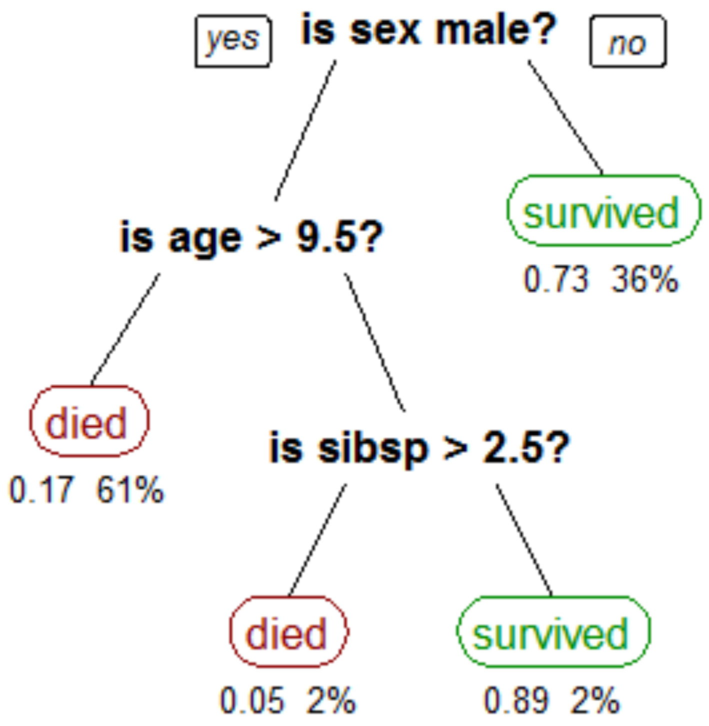


Figure 3 (b). A Logistic Regression model. Source (https://en.wikipedia.org/wiki/Logistic_regression#/media/File:Exam_pass_logistic_curve.jpeg)

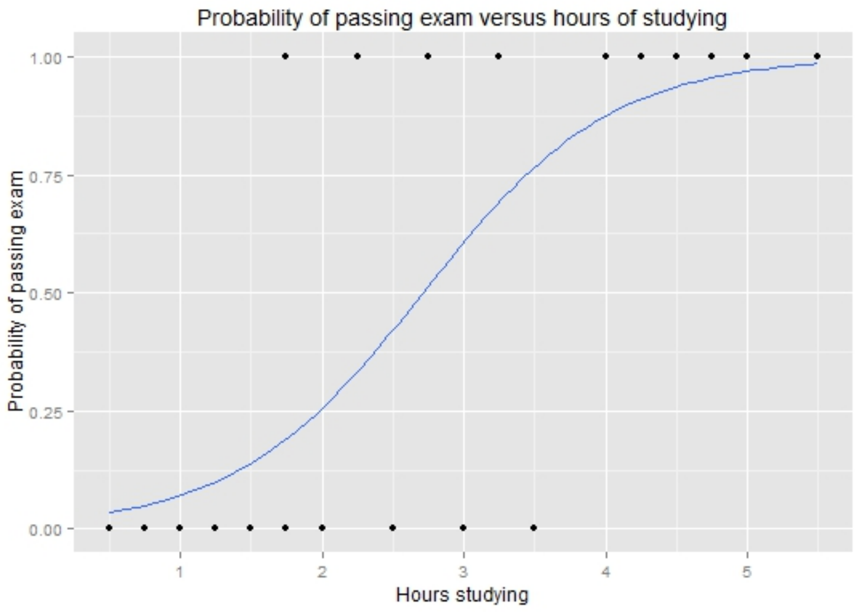


Figure 3 (c) An example of K Nearest Neighbor classification. The test case is in green. Its class label will be determined by a majority vote of its K nearest neighbors. Source: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:KnnClassification.svg

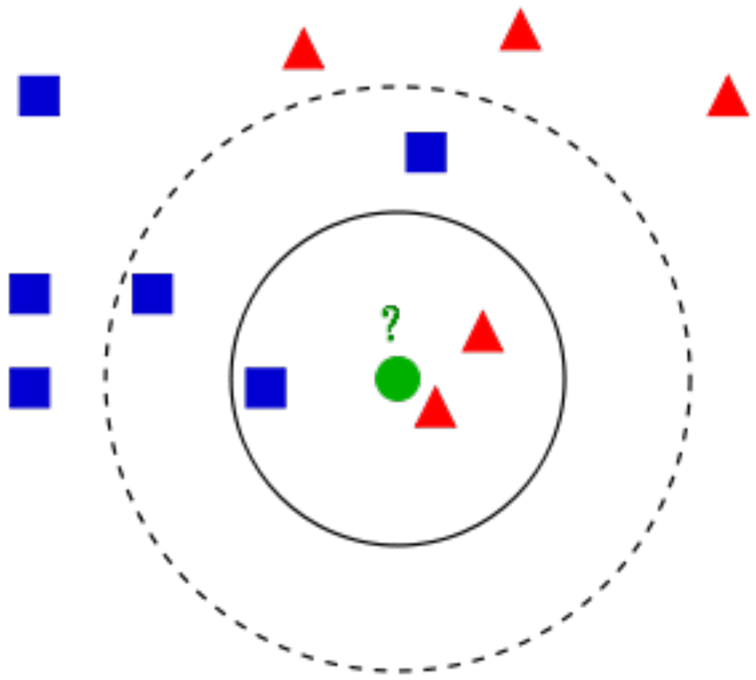


Figure 3 (d). A Support Vector Machine model for classification. Source: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

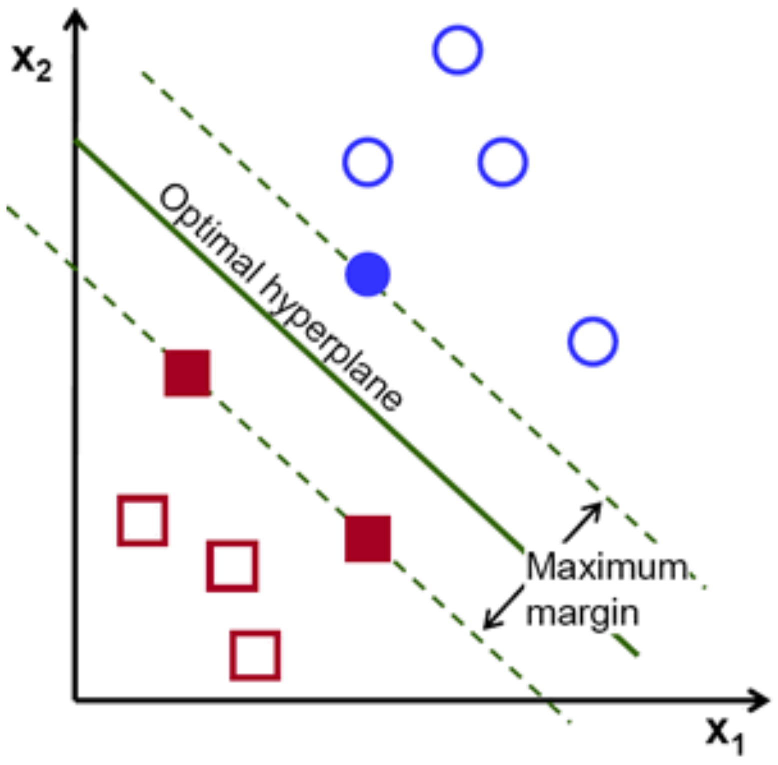


Figure 4 Distribution of market capitalization (market_cap, in billions) in the transformed dataset.

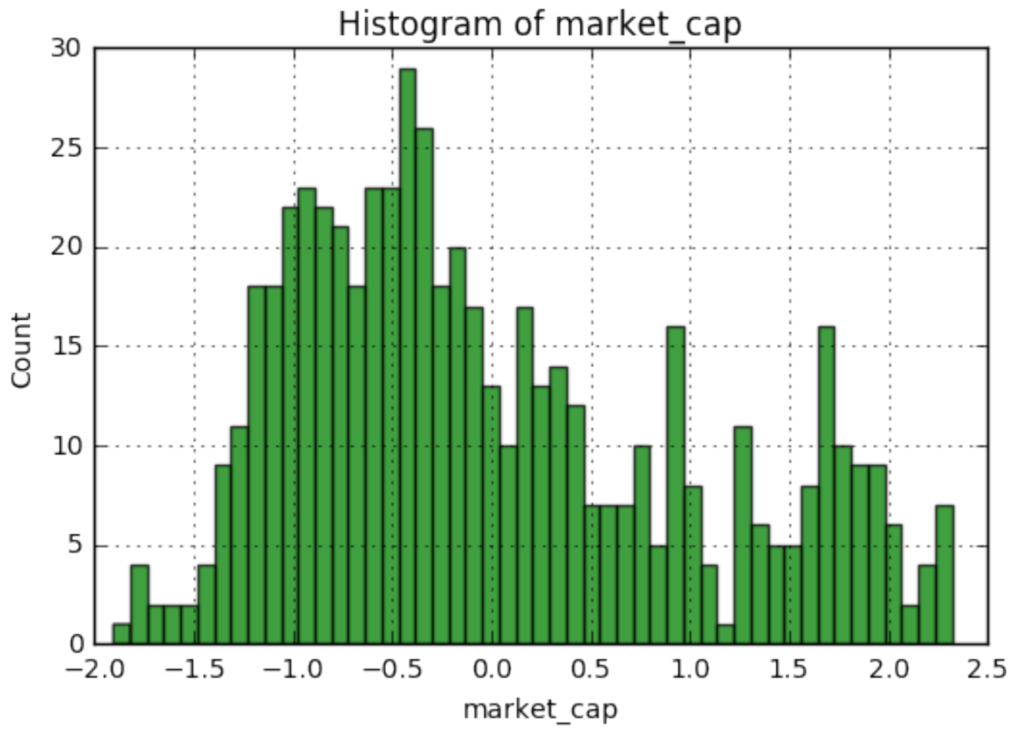


Figure 5 shows the distribution of P/E ratio (p_e) in the transformed dataset.

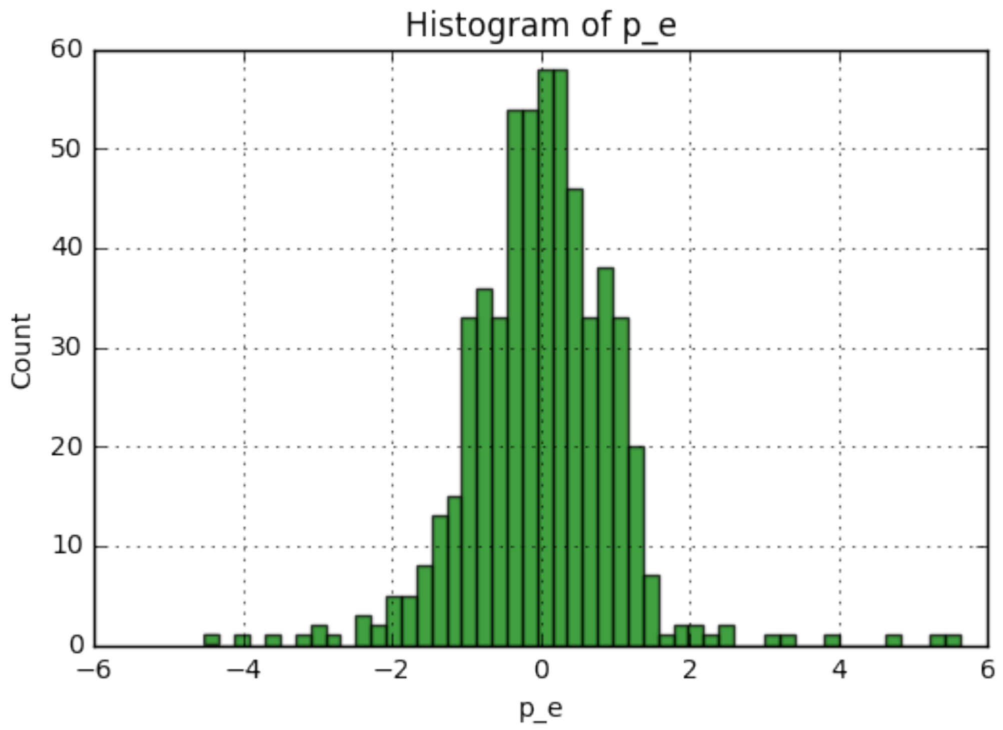


Figure 6 (a). Performance Metrics for Naïve Bayes, K Nearest Neighbors, and Decision Tree classifiers

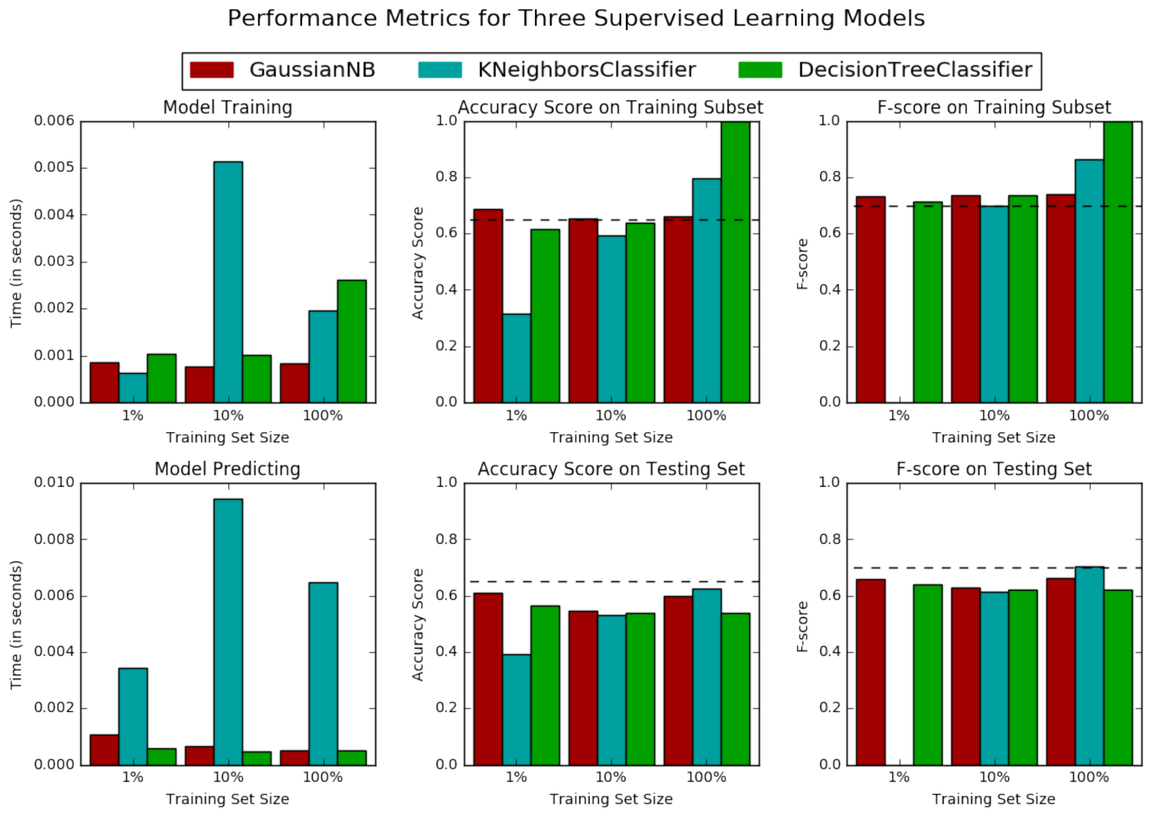


Figure 6 (b). Performance Metrics for Logistic Regression, Support Vector Machines and Random Forest classifiers

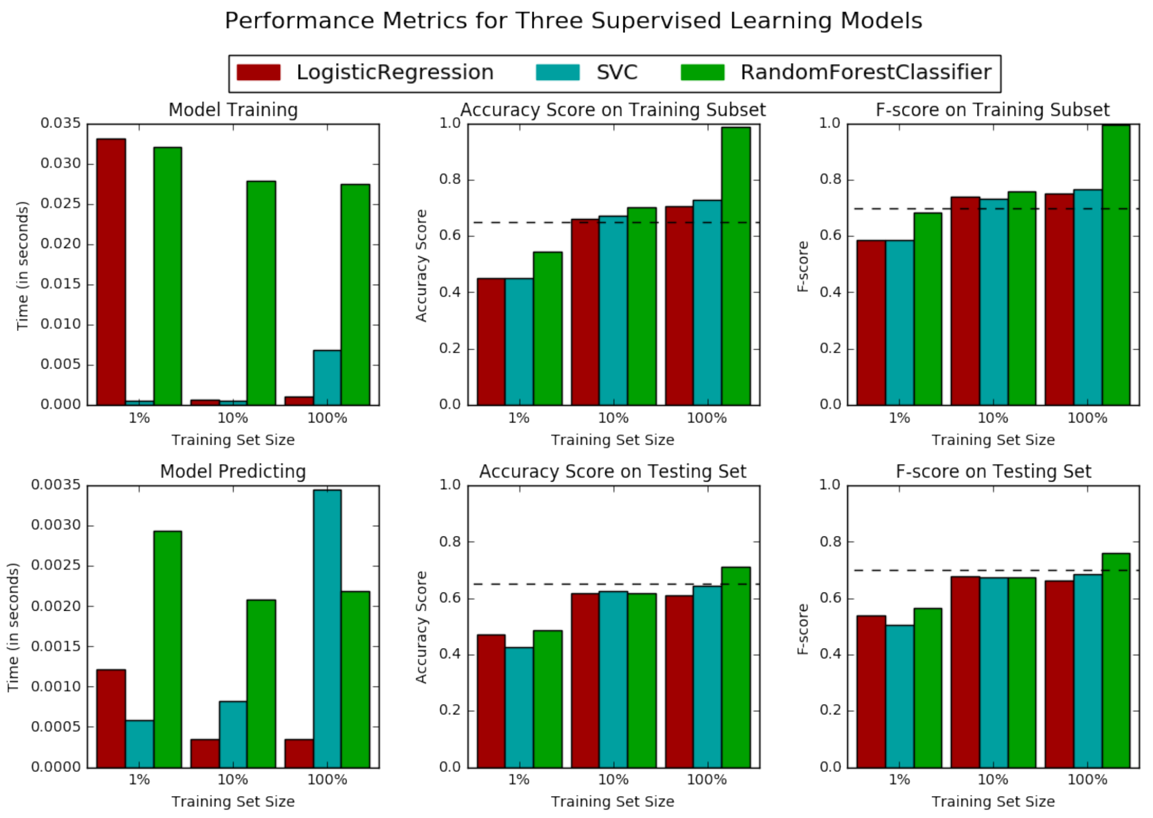


Figure 7. Five most predictive features based on the Random Forest classifier

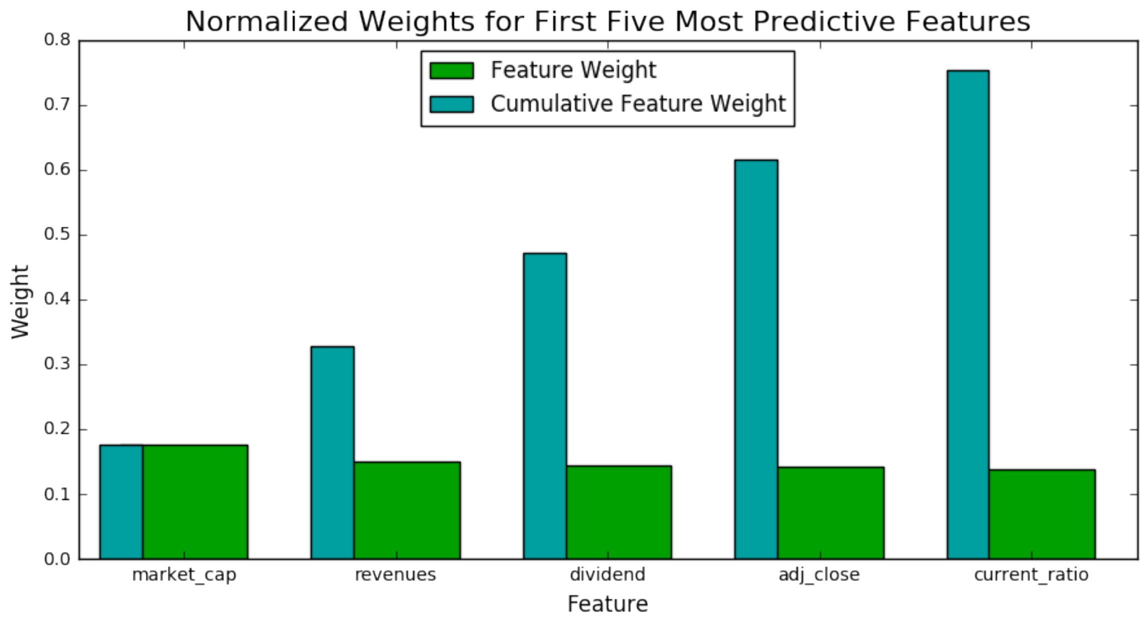
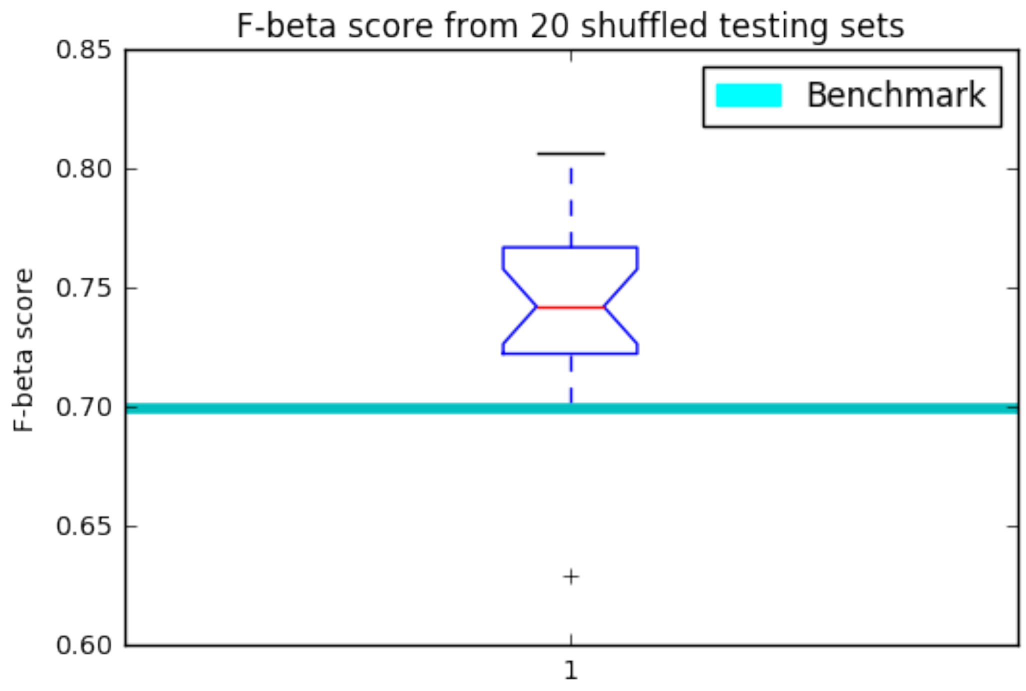


Figure 8. F-beta score from the Random Forest classifier on 20 shuffled testing sets, as compared with the F-beta score from the benchmark.


F-beta score from 20 shuffled testing sets
Benchmark
F-beta score