Using Supervised Learning Algorithms for Titanic Survival Prediction

Venkat Narra (5033)
Sai Siddhardha Maguluri (5033)

Abstract

In this project, we aim to build a machine-learning agent that predicts the survival of the passenger in the Titanic disaster. We implemented four supervised learning algorithms to check which will give better accuracy. This project requires a combination of data cleaning, exploratory data analysis, and observation using machine learning techniques like KNN, Naive Bayes, Random Forest, and Decision Tree to build an accurate model that can generalize to test data.

1. Project Domain

The Titanic data set `https://www.kaggle.com/competitions/titanic/overview` provided by Kaggle includes information on 891 passengers, with 342 of them surviving and 549 perishing. The goal is to build a predictive model to determine the likelihood of a passenger surviving the Titanic disaster based on various features such as age, gender, ticket class, fare, and cabin. Since we need to predict whether a passenger survived the disaster or not, we will have only 2 prediction classes(0 and 1). So, this is a classification task.

2. Learning Methods

In this project, we implemented four supervised learning algorithms namely - Naive Bayes, K-Nearest Neighbors(KNN), Decision Tree, and Random Forest.

2.1. Gaussian Naive Bayes

The Gaussian Naive Bayes algorithm is a classification algorithm that is based on Bayes' theorem. The algorithm assumes that the features are independent of each other given the class label.

The training process involves calculating the mean and variance of each feature for each class and also calculating the prior probabilities of each class. The prior probability of a class is calculated as the count of instances of that class divided by the total number of instances in the training data. The conditional probability of a feature given a class is calculated as the count of instances where the feature takes a particular value and the class is a particular value, divided by the count of instances where the class takes that particular value.

The predict function uses the trained model to predict the target labels for the given data. For each feature vector, it calculates the posterior probabilities for each class using the Naive Bayes formula. The class with the highest posterior probability is then selected as the predicted class for that feature vector.

The Naive Bayes formula used to calculate the posterior probability of a class given a feature vector is shown below:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, ..., x_n)} \quad (1)$$

where P(y—x1, x2, ..., xn) is the posterior probability of class y given feature vector (x1, x2, ..., xn), P(y) is the prior probability of class y, P(xi—y) is the conditional probability of feature xi given class y, and P(x1, x2, ..., xn) is the marginal probability of the feature vector.

The prior probability of a class is given by:

$$P(y = c) = \frac{count(y = c)}{n} \quad (2)$$

where count(y=c) is the number of instances in the training data where the class is c, and n is the total number of instances in the training data.

The conditional probability of a feature given a class is given by:

$$P(x_i|y = c) = \frac{count(x_i, y = c)}{count(y = c)} \quad (3)$$

where count(xi, y=c) is the number of instances in the training data where the feature xi takes a particular value and the class is c, and count(y=c) is the number of instances in the training data where the class is c.

The predicted class for a feature vector (x1, x2, ..., xn) is given by:

$$\hat{y} = argmax_{c \in C} P(y = c) \prod_{i=1}^{n} P(x_i | y = c) \quad (4)$$

where C is the set of all possible classes, P(y=c) is the prior probability of class c, and P(xi—y=c) is the conditional probability of feature xi given class c. The argmax function selects the class with the highest posterior probability as the predicted class.

2.2. K-Nearest Neighbors (KNN)

The KNN algorithm is a non-parametric classification algorithm that assigns a label to a data point based on the labels of its nearest neighbors in the training data. The number of neighbors to consider is a hyperparameter, denoted by K. Below is an implementation of the KNN algorithm.

In the KNN implementation, the fit method is used to train the model on the training data by storing the training data and their corresponding labels. The predict method is used to predict the labels of the test data.

To predict the labels of the test data, the Euclidean distance between each test data and all the training data is calculated using the formula below:

$$distance = \sqrt{\sum_{i=1}^{n} (x_{test,i} - x_{train,i})^2} \quad (5)$$

The find n_neighbors method is used to find the K nearest neighbors of a given test data point. The K nearest neighbors of each test data point are then identified by finding the indices of the K smallest distances using argsort function from numpy package. The most frequent label among the K nearest neighbors is assigned as the predicted label for the test data point.

2.3. Decision Tree

A decision tree is a machine learning algorithm used for both classification and regression tasks. The algorithm creates a tree-like model of decisions which enables the algorithm to make predictions or classify new instances based on a set of features. We choose to implement Decision Trees because they can handle both categorical and numerical data. The construction of a decision tree involves recursively partitioning the dataset based on the values of different features. At each step, the algorithm selects the best feature to split the data, aiming to maximize information gain or decrease impurity measures such as entropy or the Gini index. The decision tree learning process continues until a stopping criterion is met, such as reaching a maximum depth or a minimum number of instances in a leaf node. For our algorithm, the stopping criterion we choose a default maximum depth of 3, and the minimum number of instances in a leaf node is 2.

Depending on the splitting criteria, there are different kinds of decision algorithms. They are ID3(Iterative Dichotomiser 3), C4.5, and CART(Classification and Regression Tasks). ID3 uses entropy as the splitting criterion and selects the feature with the highest information gain at each step. CART uses the Gini index as the default impurity measure for splitting, but it can also employ other measures like entropy. We implemented both ID3 and CART algorithms. Based on user criteria like 'entropy' or 'gini', we calculated entropy and gini index respectively.

2.4. Random Forest

Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. It creates a set of decision trees using random subsets of the training data and random subsets of features. Once all the decision trees are constructed, predictions are made by combining the predictions of each individual tree. For classification tasks, the class that receives the majority of votes from the trees is chosen as the final prediction. Since Decision Trees tend to over-fit the data, we choose to implement Random Forest for better prediction.

3. Literature Review

K-nearest-neighbor classification was to execute characteristic analysis when clear parametric approximations of probability densities were unknown or difficult to determine. In 1951, Fix and Hodges introduced a non-parametric algorithm for pattern classification that has since become known as the K-nearest neighbor rule. KNN is generally used for classification problems. [1] highlights the KNN algorithm and presents the overview of 10 different variants of KNN such as Locally Adaptive KNN, Weight Adjusted KNN, Adaptive KNN, and so on. KNN becomes significantly slower as the volume of data increases making it an impractical choice in environments where predictions need to be made rapidly. However, faster algorithms are developed [2]. Yonghong and Jain[7] use a Naive Bayes classifier to classify the text document and mentioned it outperformed the other two classification methods(KNN, Decision trees) on their data sets.

Decision Tree is a non-parametric and can efficiently deal with large, complicated datasets without imposing a complicated parametric structure. paper[3] introduces frequently used algorithms used to develop decision trees (including CART, C4.5, CHAID, and QUEST) and also mentioned the common usages of decision tree models. Random Forests is an ensemble learning algorithm that generate many classifiers and aggregate their results. Two well-known methods

of ensemble methods are boosting and bagging [5] of classification trees. In bagging, successive trees do not depend on earlier trees, each is independently constructed using a bootstrap sample of the data set. In the end, a simple majority vote is taken for prediction.

4. Hypotheses

1. Random Forest gives better accuracy than a decision Tree.

2. The splitting criteria used in a decision tree algorithm can have an effect on the accuracy rate of the resulting model. We hypothesize that the Gini index is better compared to entropy as a splitting criterion because it tends to favor larger partitions with more uniform class distributions. It is effective when the class labels are well-separated in the feature space.

3. We hypothesize that the choice of the k value will have an impact on the accuracy of the KNN algorithm.

4. We hypothesize that the k-Nearest Neighbors (KNN) algorithm will outperform Naive Bayes in classification accuracy.

5. Experiments

For our experiments, we implemented four supervised learning algorithms- KNN, Naive Bayes, Decision Tree, and Random Forest.

We start with pre-processing of our data set. Firstly, We identified that few of the features don't contribute much for the algorithm to learn and predict the label. So, we dropped features the following features: PassengerId, Name, Ticket, and Cabin. Secondly, We observed that some features are missing data. The feature that is missing data is Age. So, we have taken the median of all ages and used that value to fill in the missing age. Since values in the Sex and Embarked columns are categorical values we mapped them to real values. for example, we assigned a value of 0 for the Sex column if a passenger is female, and a value of 1 for the Sex column if a passenger is male. In the same way, we used values 0,1,2 for categories of S, C, and Q of Embarked column respectively.

For the Decision Tree, the process we followed is, to calculate the information gain or Gini gain of all features and take the one with maximum gain, and split the data into two sets. For each of the sets, we repeated the same process until the stop condition is met. The stopping conditions we choose are the maximum depth of the tree being 5 and the minimum no of samples in a split being 2. Once the tree is built, for leaf nodes we assigned labels based on the majority label value(i.e. either 0 or 1). We then used the decision tree model to predict the labels for test data using predict() function. A decision tree classifier with the Gini index as a criterion gave us a better accuracy rate than an entropy one. This makes our hypothesis 2 true. We then implemented Random Forest to test out our hypothesis 1. To our surprise, Random Forest has less accuracy than the decision tree. we believe it is because of our dataset size and also because we have less number of features.

For KNN, we experimented with different k values like (1,3,5,7), and we got accuracies of 0.7622, 0.7392, 0.7902, and 0.7902 respectively. we observed that for the k values of 5,7, we got a higher accuracy. This makes our hypothesis 3 true. The experiments conducted on the Titanic dataset demonstrated that the KNN algorithm, with k=7, achieved a higher classification accuracy compared to Naive Bayes.

6. Evaluation Metrics

For this Titanic dataset, we are interested in predicting whether a passenger on the Titanic survived or not. This is a binary classification problem, where the positive class represents passengers who survived, and the negative class represents passengers who did not survive.

To evaluate the performance of a binary classification model on this dataset, we use several evaluation metrics, including accuracy, precision, recall, and F1-score.

6.1. Accuracy

Accuracy is the proportion of correct predictions out of all predictions made by the model. In the context of the Titanic dataset, accuracy represents the proportion of passengers whose survival status was correctly predicted by the model.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

where TP (True Positive) is the number of passengers who survived and were correctly predicted to have survived, TN (True Negative) is the number of passengers who did not survive and were correctly predicted to have not survived, FP (False Positive) is the number of passengers who did not survive but were incorrectly predicted to have survived, and FN (False Negative) is the number of passengers who survived but were incorrectly predicted to have not survived.

6.2. Precision

Precision is the proportion of true positive predictions out of all positive predictions made by the model. In the context of the Titanic dataset, precision represents the proportion of passengers predicted to have survived who actually survived.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

6.3. Recall

Recall is the proportion of true positive predictions out of all actual positive instances in the dataset. In the context of the Titanic dataset, recall represents the proportion of passengers who actually survived who were correctly predicted to have survived.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

6.4. F1-score

The F1-score is the harmonic mean of precision and recall and provides a balance between the two metrics. In the context of the Titanic dataset, the F1-score represents the overall performance of the model in terms of both precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{9}$$

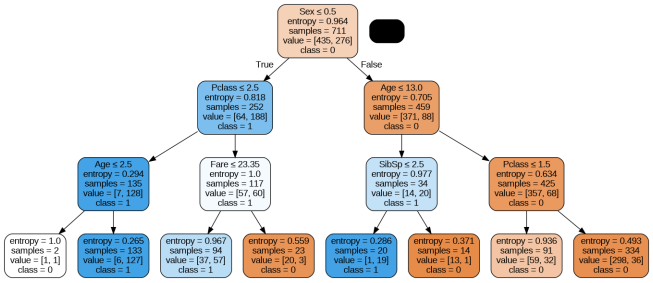
The below table provides the accuracy scores of our algorithms:

Experiment Results:

Table 1. accuracy score of the models

Algorithm	Accuracy
Decision Tree	0.8483
Random Forest	0.7752
K-NN(k=7)	0.7902
Naive Bayes	0.7342

The following image shows the constructed decision tree:



7. Conclusion and Future Work

In conclusion, we implemented various supervised learning algorithms, evaluated them using different metrics, and found out that our decision tree model has the highest accuracy among the others. In the future, we would like to perform exploratory data analysis as mentioned in [6] and also we would like to create new features from existing features that may enhance the predictive power of the model. For example, a new feature like familySize, depends on the SibSp and Parch features of the dataset. Also, we plan to implement different variants of the KNN algorithm to improve the prediction rate as our model accuracy is not up to mark.

8. Contributions

Venkat Narra implemented the K-Nearest Neighbors and Naive Bayes algorithms and Sai Siddhardha Maguluri implemented the Decision Tree and Random Forest algorithms.

References

1. K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

2. Pan, J. S., Qiao, Y. L., Sun, S. H. 2004. A fast k nearest neighbors classification algorithm. Ieice Trans Fundamentals A, 87(4), págs. 961-963.

3. Song YY, Lu Y. Decision tree methods: applications for classification and prediction. Shanghai Arch Psychiatry. 2015 Apr 25;27(2):130-5. doi: 10.11919/j.issn.1002-0829.215044. PMID: 26120265; PMCID: PMC4466856.

4. Breiman, L. (2001) Random Forests. Machine Learning, 45, 5-32. https://doi.org/10.1023/A:1010933404324

5. Breiman, L. (1996) Bagging Predictors. Machine Learning, 24, 123-140. https://doi.org/10.1007/BF00058655

6. K. Singh, R. Nagpal and R. Sehgal, "Exploratory Data Analysis and Machine Learning on Titanic Disaster Dataset," 2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence), Noida, India, 2020, pp. 320-326, doi: 10.1109/Confluence47617.2020.9057955.

7. Yonghong Li and A. K. Jain, "Classification of text documents," Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170), Brisbane, QLD, Australia, 1998, pp. 1295-1297 vol.2, doi: 10.1109/ICPR.1998.711938.