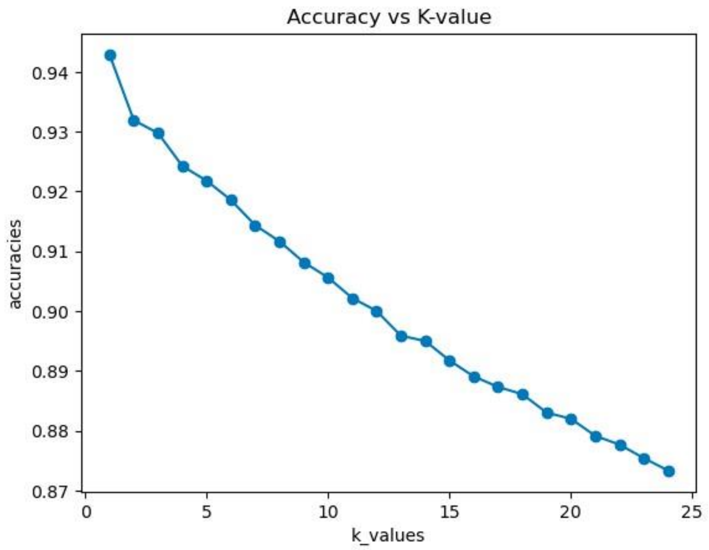
Forest Classification Project Report

By: Harsith Vemuri and Pranav Tummalapalli
Accuracy Score: 0.78

Step by Step Documentation:

1. Firstly, the train and test data were created by converting the data from the train and test CSV files into Pandas Dataframes.
2. Then in an attempt to normalize the data and in an attempt for it to fit within a gaussian curve, feature scaling was done using the z-normalization technique by the function call StandardScaler().
3. Due to the imbalanced nature of the dataset, sampling was done to balance the data.
    a. This was achieved by using SMOTE which stands for Synthetic Minority Oversampling Technique, this technique selects the minority feature values at random and generates new points along the line drawn across the minority feature values.
4. Then the project comes to the crux of the problem given, due to the sheer size of the dataset, feature selection is a vital part of the data preprocessing required for this project.
    a. Upon the grand variety of feature selection techniques, SelectPercentile, was chosen due to its high levels of accuracy when dealing with continuous data types.
        i. SelectPercentile is a feature selection algorithm that essentially selects the top k percent of the features, using a scoring function, which we deemed to be mutual_info_regression due to its flexibility in capturing complex relationships, which does not make restrictive assumptions about linearity. ii. In our code, the percentage that was chosen was 38%, this was determined through various trial runs of varying percentages in the SelectPercentile feature selction algorithm.
5. The three classifiers that were chosen were KNN, Decision Tree, and Naïve Bayes. This was due to the fact that Neural Networks although quite accurate have a very large computational time complexity.
    a. The KNN classifier was chosen due to its non-parametric fashion allowing for great adaptability to data distributions. Secondly, when feature selection and scaling is applied, the distance computations in KNN are more effective, thereby causing a more efficient classification.
        i. The KNN classifier was called by utilizing the function KNeighborsClassifier() with the parameter n_neighbors = k, which depicts the number of neighbors the KNN algorithm has to take into account for the classification. Then this classifier is cross validated and trained upon the Cross Validation method. In this method various k-values are used to train the model by using the function call cross_validate() with the parameters classifier, x (train features), y (train labels), cv = 10. The parameter classifier is the variable classifier that is being passed from the KNN algorithm. The parameter x and y are the train features and train

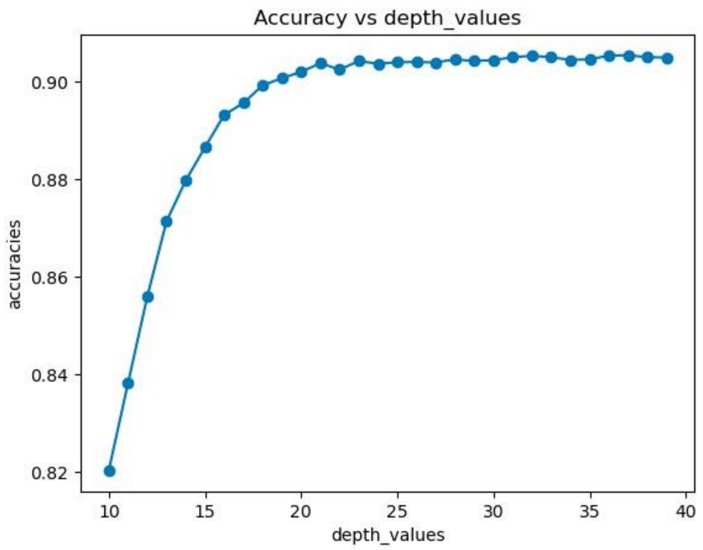
labels. And lastly, the cv = 10 refers to how many folds the crossvalidation algorithm must do. 10 folds was chosen through many trials.



ii.
iii. Shown above is an Accuracy vs K-value plot depicting the accuracies of the train data using Cross validation with varying k-values. Using the graph above the highest accuracy (0.943) happens to have a k-value of 1.

b. Secondly, Decision Trees classifier was chosen due to its ability to conduct Automatic Feature Thresholding, wherein the decision tree performs feature thresholding to split the data into homogenous subsets. In continuous datasets the classifier determines the best threshold for each feature thereby splitting the nodes optimally to maximize homogenous features in the subset.

i. The Decision Tree classifier was called by utilizing the function tree.DecisionTreeClassifier() with parameters: criterion = ‘entroy’ and max_depth = depth. The parameter criterion = ‘entropy’ defines the functions role of determining the node’s impurity (entropy) when splitting features into nodes, with the primary goal of maximizing information gain. The reason behind utilizing entropy is due to its ability to provide more balanced splits into multi-class classification problems. Additionally, entropy captures the uncertainty in the distribution of class labels, making it suitable for datasets with imbalanced class distributions. The accuracy of this model was calculated through Cross validation through the function cross_val_score with the parameters clf, X, y, cv = 10, scoring = ‘accuracy’. The parameter clf refers to the classifier (decision trees) object. The X and y are the train features and train labels. Cv = 10 refers to the number of folds in the cross-validation function. The parameter scoring = ‘accuracy’ refers the use of accuracy as a scoring function for each fold across cross validation. Benefit of using accuracy



ii.

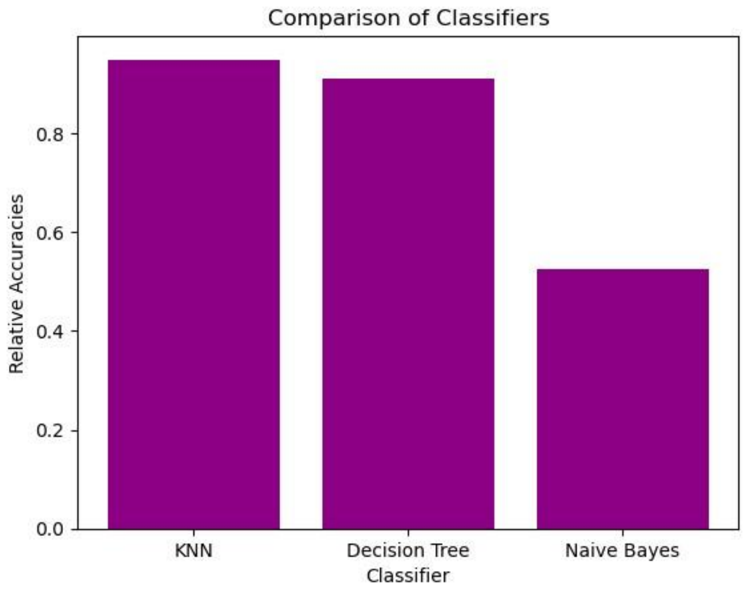
iii. Shown above is an Accuracy vs depth_values plot depicting the accuracies with respect to depth_values in a decision tree classifier model, completed by utilizing various depth values in cross validation. Using the graph above the highest accuracy (0.909) happens to have a depth value of 34.

c. Lastly, Naïve Bayes was chosen due to its rapid computational time complexity.

i. The Naïve Bayes classifier was called by utilizing the function NBClassfier which takes in parameters of the train features, test features, and labels. In the Naïve Bayes function the Gaussian Naïve Bayes type was chosen due to its high accuracy when dealing with continuous data. This method takes no parameters.

ii. The accuracy of the algorithm came out to be 0.525.

6. Comparision and Conclusion



a.

b. From the graph it is evident that the highest accuracy is from the KNN classifier. Thus, KNN was chosen as the classifier to predict the target labels of the test dataset.