Classification Comparison Report by Nikhil E

Introduction:

Upon reviewing the dataset, we identified a dataset comprising 11 columns with voter details, such as state, voter ID, gender, family, political party, age, salary, and voting history, denoted by columns D, F, and G. However, we have concluded that certain columns, namely D, F, and G, contain irrelevant data and will be omitted from further analysis. Our primary focus will be on selecting the pertinent independent variables to predict the dependent variable.

Data Pre-processing:

Moreover, we conducted data reduction by eliminating three irrelevant columns (D, F, and G) from the dataset due to their lack of significance and relevance to the remaining data. This reduction resulted in a modified dataset, where columns 1 to 8 were considered significant, while columns 9 to 11 were deemed insignificant. Additionally, we addressed any missing values (NA) during this process.

To initiate the analysis, we performed data transformation on the "voted last election" column, converting binomial values to a "0 or 1" format. This step is crucial for maintaining consistency in our analysis and model-building efforts.

Upon closer examination, we observed that certain variables in our dataset are categorical, including "State," "Gender," "Political Party," and "Voted_Last_Election." To ensure accurate interpretation by our models, we needed to encode these variables appropriately.

We converted "State," "Gender," and "Political Party" into factors, treating them as distinct and separate categories. This approach allows our models to identify and differentiate between the various states, genders, and political parties present in the dataset. Conversely, for the "Voted_Last_Election" variable, we encoded it as a binary factor with two levels, '0' and '1.' In simple terms, it indicates whether a person participated in the most recent election or not.

These encoding processes enabled us to refine our dataset, creating a more focused and meaningful set of variables for our analysis.

Data Splitting:

Now, let's proceed with data splitting. To efficiently train and evaluate our models, we utilized the "createDataPartition" function from the caret package. This function facilitated the division of the dataset into a training set and a test set. This random split ensured that the machine learning model was trained on one subset of the data and tested on another, unseen portion. Consequently, this approach allowed us to assess the model's ability to generalize and make accurate predictions. Our dataset underwent a division into an 80% training set and a 20% test set, establishing a balanced and reliable method for measuring the model's performance.

Evaluation models:

With consideration for an individual's age and income, our goal is to enable the computer to predict whether they will participate in the upcoming election. To accomplish this, we employ a range of techniques, including Logistic Regression, Support Vector Machine, K-Nearest Neighbors, Decision Tree, and Random Forest, to instruct the computer in making these forecasts.

Let’s understand each of the algorithms mentioned above:

1. Logistic Regression:

Logistic regression is widely employed in binary classification, calculating the probability that an individual case falls into a specific category. In our context, the target variable is the binary vote outcome, while age and salary serve as key determinants.

Classification Comparison Report by Nikhil E

2. Support Vector Machine (SVM):
SVM proves effective in classification tasks by identifying the optimal hyperplane, a decision boundary separating two classes. In our scenario, SVM seeks a hyperplane to efficiently distinguish individuals based on age and income, categorizing them into voters and non-voters.

3. K-Nearest Neighbors (KNN):
Known for its simplicity and efficiency, KNN organizes a new data point by considering its k nearest neighbors. To assess proximity, the establishment of a distance metric, such as Euclidean distances, is crucial.

4. Decision Tree:
Decision Trees construct a tree-like structure by iteratively dividing data based on features. Each inner node evaluates a specific feature, with outcomes depicted through branches. The final class is represented by leaf nodes, making decision trees interpretable as non-linear relationships.

5. Random Forest:
Operating as an ensemble method, Random Forest builds multiple decision trees and combines their predictions to enhance overall accuracy.

Now, let's look at how each model through code:

Logistic Regression:

The line "train(...) - LogRegModel" initiates the creation of a LogRegModel, which learns from the training set. This model delves into understanding how age and income may influence one's likelihood of voting in the previous election. It employs the "glm" or Generalized Linear Model technique, considering two potential outcomes: either voting (1) or not voting (0).

```
# Log_reg_pred = predict(...)
# After training, the model is utilized to predict whether each user in the test_group voted or didn't vote based on age and salary.

# CONFUSIONMATRILE(...) - LOG_REG_cm
# This line compares the model's predictions against the actual results in the test_set, generating a confusion matrix as a report card for the model.

# Log_reg_cm_overall <- Log_reg_accuracy["Accuracy"]
# This line calculates the accuracy of the model, representing the percentage of correct answers. A higher accuracy value indicates better performance.

# log_reg_cm$byClass["Precision"] <- log_reg_precision
# Precision measures the proportion of voters correctly identified by the model, answering the question of how often the model correctly predicts someone voted.

# Log_reg_c$byClass["Recall"] <- Log_Reg_recall
# Recall quantifies the number of voters the model successfully identified, akin to asking how many people the model found among those who voted.

# Log_reg_cm = byClass["F1"], F1 = Log_Reg_f1
# The F1 score is calculated by combining recall and precision, providing an accurate representation of the model's performance.

# Accuracy: It is the percentage of votes that the computer predicts correctly.
# Precision: It is the number of times the machine predicts votes correctly.
```

Classification Comparison Report by Nikhil E

Recall: It is the number of actual voters that the machine successfully identifies.
F1 Score: F1 Score is the balance between accuracy and recall.

roc(...) - log_reg_roc
The ROC curve is generated by comparing the test set results with the model predictions, assessing the model's ability to differentiate between voters and non-voters.

auc(log_reg_roc)
AUC stands for Area Under the Curve, measuring the model's overall performance. AUC distinguishes between voters (AUC = 0.00) and non-voters (AUC = 1.00).

In summary, this section of the code involves training a model to estimate a person's voting probability based on age and income, evaluating the model's performance, and assessing its ability to differentiate between voters and non-voters.

SVM:

This code constructs an SVM model named svm_model, akin to the previous logistic regression function. Utilizing the svmRadial method, it employs a support vector machine (SVM) to learn from the data in the training_set, discerning how an individual's age and income may influence their likelihood of voting in the recent election.

Upon training the SVM model, it becomes capable of forecasting whether another group in our test set will vote by examining the age and salary variables within that group. The resulting predictions are stored in the SVM_pred variable.

The subsequent section of the code assesses the performance of the SVM model, mirroring the evaluation process employed for logistic regression models. It gauges how well the predictions align with the actual outcomes in the test set, calculating metrics such as recall rate, accuracy rate, precision rate, and F1 score to comprehensively understand various facets of the model's performance. Similar to the logistic regression model, this section scrutinizes the SVM's proficiency in distinguishing between voters (i.e., "voter") and non-voters. Additionally, it generates an ROC curve for the SVM and computes the Area Under the Curve (AUC) to quantify the model's discriminatory power.

In essence, this segment of the code establishes a model to forecast voting behavior based on factors like age and salary. Subsequently, it assesses the model's predictions and measures its effectiveness using diverse metrics.

Creating cross-validation folds:

In this process, ten different groups or folds are defined based on the dataset named "training_set." The model is both trained and evaluated against various data subsets or folds. This approach ensures that the model undergoes assessment using multiple subsets, enhancing its overall evaluation. A grid of potential values for the parameter "k" is created. Here, "k" signifies the number of neighbors the model considers when making predictions in the k-nearest neighbors (KNN) context. The model is tested and trained using a range of values for 'k' (from 1 to 20) to identify the optimal value.

In this specific example, cross-validation, denoted as "cv," is employed. Cross-validation involves splitting the dataset into several folds or groups, and the model is tested on some folds while being trained on others (referred to as "folds"). The generated folds are indicated by the index argument. The code utilizes the train function to train a k-nearest neighbor (kNN) model, with age and salary from the training_set serving as predictors for predicting the "voted_last_election" variable. With the technique set to "on," tuneGrid specifies

Classification Comparison Report by Nikhil E

the grid of values for "k" to be tried during the training phase. The configuration, defined by trControl, manages the cross-validation process.

The line then returns the best-performing parameter "k" from the KNN model's tuning results after completing the training phase. This step is akin to determining the optimal number of neighbors for achieving the most accurate predictions on the training data. In summary, this code prepares and trains a KNN (k-nearest neighbor) model, aiming to find the optimal value for the 'k' parameter, representing the number of neighbors considered during the prediction process.

KNN:

This code trains a k-nearest neighbor (kNN) model using a dataset to forecast individuals' voting behavior in the upcoming election based on factors such as age and income. The model is trained utilizing the optimal value of the parameter "k" (number of neighbors to consider), as determined earlier. Employing a distinct dataset for testing, the model is then utilized to generate predictions. Subsequently, the model's predictions are compared against the actual voting outcomes. The code assesses the model's performance by computing metrics such as model accuracy, precision, recall, F1 score, and AUC. Additionally, the code calculates the Area Under the Curve (AUC) and constructs the Receiver Operating Characteristic (ROC) curve to gauge the model's ability to discriminate voting behavior based on age and salary.

Decision Tree:

In this code, a training dataset is utilized to train a decision tree model (dt_model) designed to predict whether individuals voted based on their age and income during the preceding election. The Decision Tree model is constructed as a tree structure using the "rpart" method, with decisions guided by specific attributes. The trained model is then applied to forecast voting outcomes for a separate test dataset. Subsequently, the code assesses the model's performance by comparing its predictions to the actual voting results, employing metrics such as accuracy, precision, concordance, and F1 score. Additionally, the algorithm calculates the AUC (Area Under the Curve) and generates an ROC (Receiver Operating Characteristic) curve to evaluate the model's ability to distinguish between voters (those who voted) and non-voters. Collectively, these metrics offer a comprehensive perspective on the decision tree model's effectiveness in predicting voting behavior based on age and income information.

Random Forest:

Using a training dataset, this code trains a rf_model, which is a Random Forest model, to predict individuals' voting behavior based on their age and income in the previous election. The Random Forest technique employs an ensemble approach, constructing multiple decision trees and consolidating their predictions. Subsequently, the trained model is applied to another test dataset to forecast voting outcomes. The code assesses the model's performance by computing metrics such as precision, accuracy, recall, and F1 score, comparing the predictions against the actual voting results.

Moreover, the algorithm calculates the Area Under the Curve (AUC) and generates a Receiver Operating Characteristic (ROC) curve. These steps aim to evaluate how effectively the Random Forest model predicts voting behavior. The comprehensive set of indicators provides a thorough assessment of the model's ability to distinguish between voters and non-voters, leveraging the collective strength of multiple decision trees based on age and salary features.

Comparing the models:

Classification Comparison Report by Nikhil E

Now, we assess the predictive performance of each method by considering accuracy (representing the number of correct predictions), precision (reflecting the ability to accurately predict actual votes), recall (indicating the capability to identify all voters), F1 score (a composite metric combining precision and recall), and ROC (an additional measure evaluating model quality).

Model Performance:

Now, we will have a look at the performance of each model in detail:

Accuracy:

- Logistic Regression: 0.5045045
- SVM: 0.5145145
- KNN: 0.4864865
- Decision tree: 0.5185185
- Random Forest: 0.4894895

Precision:

- Logistic Regression: 0.5147453
- SVM: 0.5200000
- KNN: 0.5035461
- Decision tree: 0.5316901
- Random Forest: 0.5065913

Recall:

- Logistic Regression: 0.7427466
- SVM: 0.8046422
- KNN: 0.5493230
- Decision tree: 0.5841393
- Random Forest: 0.5203095

F1:

- Logistic Regression: 0.60880760
- SVM: 0.6317388
- KNN: 05254394
- Decision tree: 0.5566820
- Random Forest: 0.5133588

ROC:

- Logistic Regression: 0.4958546
- SVM: 0.5039808
- KNN: 0.4842051
- Decision tree: 0.5161360
- Random Forest: 0.4883705

Based on the assessment measures, the best model for this specific dataset is the Logistic Regression model. It suggests that it can categorize voters from the previous election by skilfully balancing memory and precision. Its overall accuracy and F1 score are also rather good.

Classification Comparison Report by Nikhil E

The ability of the Logistic Regression model to produce results that are simple to understand is what makes it so important. It can be helpful to understand how much an individual's age and salary contributed to their likelihood of voting in the last election. This information could be helpful for future political campaigns that want to emphasize demographics.

Visualization:

Finally, we generate a user-friendly bar chart to illustrate the effectiveness of each approach. The bar's height indicates the computer's ability to predict whether an individual will vote, with greater height signifying better prediction capabilities. In essence, this code instructs on predicting a person's likelihood of voting based on factors like age and income. It guides the computer in determining the most effective prediction approach.