**Machine Learning Q1 Proposal:**

**Stroke Predictive Model**

**Team Members:**

**Chetan Maviti, Kanishk Sivanandam**

**10/23/2024**

**Table of Contents**

1. **Statement of Data Mining Goal 3**
2. **Description of Dataset 3**
3. **Data Preprocessing Procedure 4**
4. **Attribute Selection and Model Classifiers Used 7**
5. **Results and Evaluation 14**
6. **Conclusion and Reproducing our Model 24**
7. **Team Members and Tasks Performed 25**
8. **References 26**

# 

# 

# 

# **Part 1 – Statement of Data Mining Goal**

Strokes are medical emergencies that require rapid and accurate diagnosis to ensure effective treatment. The sooner a stroke is recognized and cared for, the better the outcome is likely to be. Currently, the process of diagnosis of a stroke in clinical settings can be enhanced by predictive modeling, which can help in identifying potential stroke incidentes before they become critical.

The goal of our project is to develop a machine learning model capable of detecting whether or not a patient is at risk of experiencing a stroke by analyzing a host of patient health and lifestyle features from a stroke dataset. This dataset includes detailed patient information such as patient age, gender, cholesterol levels, blood pressure, etc. Utilizing this data, after necessary preprocessing to deal with missing values, inconsistent data types, etc., a predictive model is created to best identify whether or not a given individual is at risk for stroke or not.

# **Part 2 – Description of Dataset**

The stroke dataset consists of 12,760 instances, each characterized by 27 features that detail various health and lifestyle information of the patients. Data for each patient were collected on different days over a span from January 1, 2020, to December 31, 2024 (4 year span). The dataset contains a mix of binomial, qualitative, and quantitative attributes.

The attributes in the dataset are relatively uniformly distributed. The class variable, “diagnosis”, categorizes patients into “Stroke” and “No Stroke” which indicates whether or not they are at risk of stroke. This class variable is almost perfectly balanced/uniformly distributed with 50.27% labeled “Stroke” and 49.73% labeled “No Stroke”. It is also important to note that the dataset does in fact have a significant number of missing values, totaling 6,663. Attributes such as patient age, BMI index, and symptoms have at least 10% of their data missing. Still, there are no attributes with more than 17% missing values.

Here are our dataset’s attributes laid out, before any preprocessing:

* Patient\_Name: Name of the patient
* Patient\_ID: Unique ID of the patient
* Patient\_Age: Age of the patient
* Patient\_Gender: Gender of the patient
* Record\_Date: Date of recording of attributes
* Dietary\_Habits: Vegetarian, Vegan, etc.
* LDL\_Cholesterol: LDL cholesterol level (bad cholesterol)
* Work\_Type\_of\_patient: Profession-type of patient
* Metabolic\_Equivalent\_of\_Task\_Score: Measurement of physical activity
* Marital\_Status: married, single, etc.
* Physical\_Activity: Self-reported physical activity level
* Cholesterol\_Levels: Categorized activity level
* Stress\_Levels: Self-reported stress levels
* Average\_Glucose\_Level: Average glucose levels
* Heart\_Disease: Indicator for hypertension (1 = Yes, 0 = No)
* Body\_Mass\_Index: BMI of patient
* Alcohol\_Intake: Frequency of alcohol consumption
* HDL\_Cholesterol: HDL cholesterol level (good cholesterol)
* Hypertension: Indicator for hypertension (1 = Yes, 0 = No)
* Family\_History\_of\_Stroke: Family history of stroke (Yes/No)
* Diagnosis: Whether the patient is at risk for stroke or not
* Residence\_Type: Urban or rural residence
* Systolic\_BP: Systolic blood pressure
* Smoking\_Status: Current smoking status
* Diastolic\_BP: Diastolic blood pressure
* Stroke\_History: Previous stroke incidents
* Symptoms: Reported symptoms (confusion, seizures, etc.)
* Blood\_Pressure\_Levels: Combined BP levels (Systolic / Diastolic)

Note that “Diagnosis” will be used as the class variable.

# **Part 3 – Data Preprocessing Procedure**

**3.1 Symptoms Attribute Clean Up**

Before we could perform necessary preprocessing steps such as missing value replacement, we first had to deal with the unique “Symptoms” attribute. Unlike the other attributes, “Symptoms” contained lists of values rather than a single value for each instance. This list of values contained any combination of 10 symptoms listed for the patient: “Blurred Vision,” “Seizures,” “Difficulty Speaking,” “Weakness,” “Confusion,” “Headache,” “Dizziness,” “Severe Fatigue,” “Loss of Balance,” and “Numbness.” Due to this unique feature of various nominal values, we decided to split the “Symptoms” column into 10 different binary columns—one column for each symptom. This way each symptom could easily be taken into account by our classifier models without confusion. While at this step, we also removed all instances with missing values for the “Symptom” attribute, as filling in symptoms could lead to misdiagnosis. This step was accomplished on google colab using the pandas library.



**3.2 Clean Up Unnecessary Columns**

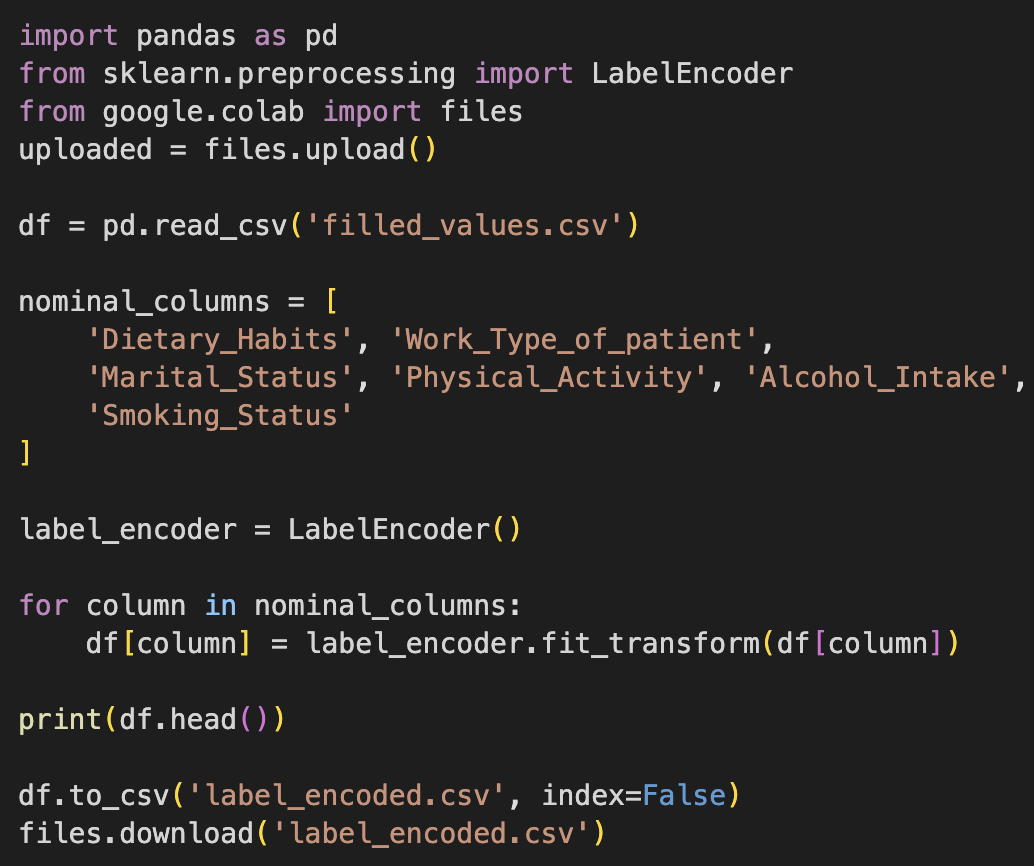
Attributes such as “Patient\_name”, “Patient\_id”, and “Record\_date” only exist to help identify patients; they are irrelevant when trying to predict future strokes and were consequently removed. Additionally, features like “cholesterol\_levels” and “blood\_pressure\_levels” were removed as they were simply compositions of two other attributes already present within the dataset (cholesterol\_levels is the same as the ldl\_cholesterol and hdl\_cholesterol attributes combined).

**3.3 Clean Up Missing Values**

Missing values are detrimental when building a model. To handle missing values, we first determined that any instance with an empty blood pressure (systolic and diastolic) attribute could be removed, as blood pressure data is extremely significant in stroke detection, and filling in this data could bias results. For the remaining attributes with missing values, we used Weka’s ReplaceMissingValues filter to fill in missing data, which replaces data with mean for numeric data and mode for nominal data. This was possible as no attribute had any more than 17% missing values, and every feature had a uniform distribution of values. Finally, we used the floor function with the MathExpression Weka filter to round down all numeric values to integers, which only affected the newly filled in values.

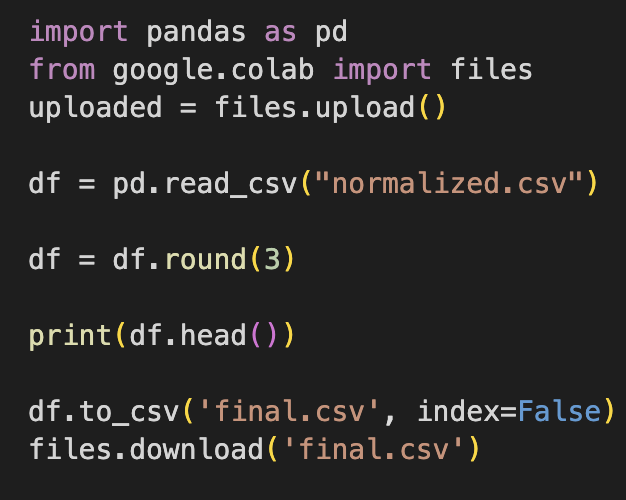
**3.4 Label Encoding**

After cleaning up the dataset, the next step was to convert the nominal attributes with more than 2 possible values (not binomial) to numeric so that they could be analyzed. This was achieved using the method of label encoding, where each possible value for an attribute is mapped to a numeric value. For example, the “Dietary\_Habits” feature, which originally contained 7 possible string values such as “Pescatarian” was converted to a numeric attribute with the numbers 0 through 6, each mapped to one of the original strings. Label encoding was carried out using sklearn’s LabelEncoder method on Google Colab.



**3.5 Data Normalization**

After the previous preprocessing steps, the different features had many different ranges of values. Some were binary, while others like the blood pressure attributes were in the hundreds. This discrepancy could overstate the importance of certain attributes when compared to others, causing issues for our classification model. Due to this, we normalized our data on the 0-1 scale using the Normalize filter in Weka. The normalization resulted in float values with a large number of decimal places, which we trimmed to just 3 places using a script in google colab.



# **Part 4 – Attribute Selection and Model Classifiers Used**

Our final dataset contained a total of 32 attributes. Having a large number of features when performing classification can lead to overfit models, therefore attribute selection must be carried out to reduce dimensionality. We used 4 attribute selectors from Weka, along with our personal choice for a fifth attribute selector.

**4.1 Attribute Selection**

**OneR:**

The OneR attribute selection algorithm creates one ‘set of rules’ to determine what attributes are worth keeping. Attributes are ranked based on their ability to classify instances using a simple set of rules derived from the dataset. The OneR algorithm is detailed below in pseudocode (Sayad):

For each feature,

For each value of that feature, make a rule as follows;

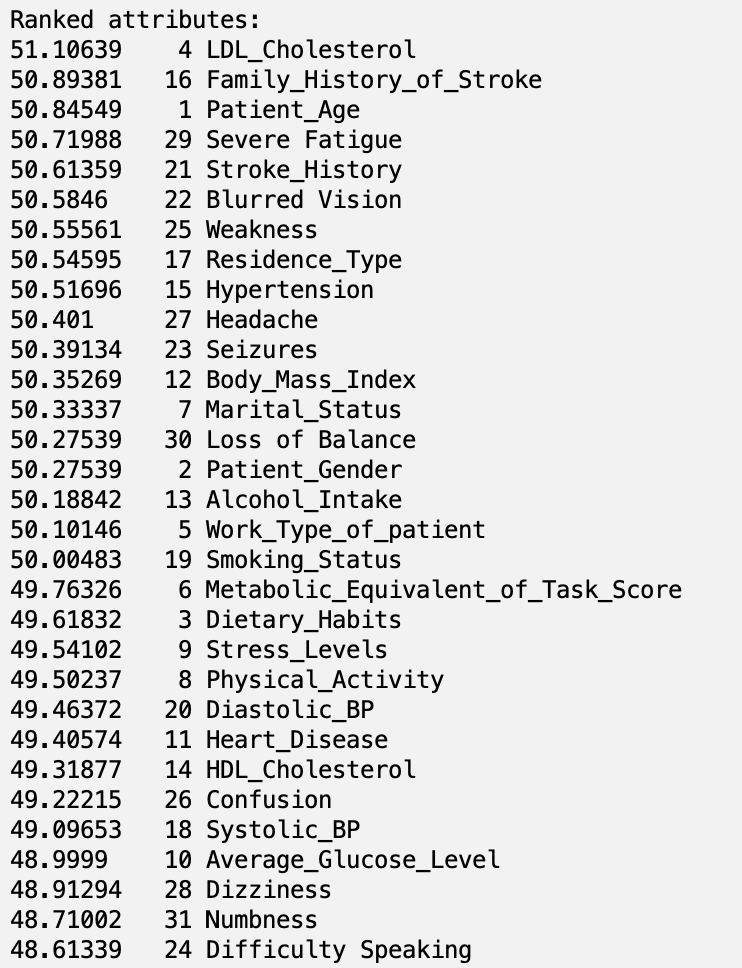
Count how often each value of target (class) appears

Find the most frequent class

Make the rule assign that class to this value of the feature

Calculate the total error of the rules of each feature

Choose the feature with the smallest total error.

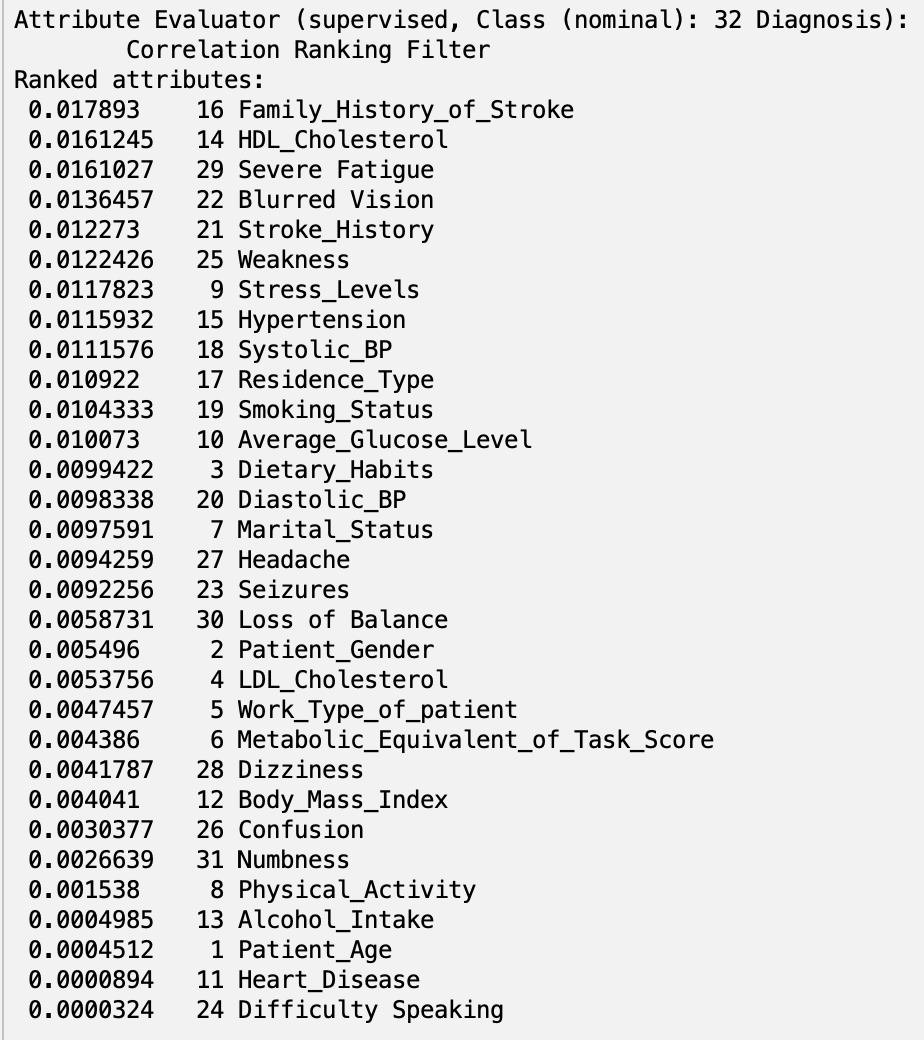


After conducting OneR in Weka, the results are shown to the right:

We chose a cut-off score of **50.55** with the minimum bucket size being 6, so the only selected attributes were 25, 22, 21, 29, 1, 16, and 4.

**CorrelationAttributeEval:**

The CorrelationAttributeEval function evaluates attributes by measuring the Pearson’s correlation between each attribute and the class. For nominal attributes, it treats each category as a separate indicator and calculates an overall correlation using a weighted average of these indicators. This correlation approach identifies attributes with strong correlation to the class. The results of the CorrelationAttributeEval for the stroke dataset are shown below:



The results of CorrelationAttributeEval are shown here:

By implementing a cut-off value of **0.01**, we are able to select only the attributes 10, 19, 17, 18, 15, 9, 25, 21, 22, 29, 14, and 16.

**PrincipalComponents (PCA):**

In a statistical sense, principal component analysis allows users to “summarize the information content in large data tables by means of a smaller set of ‘summary indices’ that can be more easily … analyzed” (Sartorius). In machine learning application, these “summary indices” are just “linear combinations or mixtures of the initial variables” (Jaadi) that will be used as attributes. These component vectors (linear combinations of initial variables) are ranked based on how much of the variance from the dataset they capture (Jaadi). The PCA of the stroke dataset is shown below:



A cut-off value of **0.80** was chosen, meaning any component vectors that accounted for less than 80% of the variance in the dataset were not used as attributes. This meant only the top 4 component vectors were utilized as attributes with PCA.

**ReliefF:**

ReliefF is an attribute selection algorithm that evaluates the importance of features by analyzing how effectively each attribute differentiates between neighboring instances from different classes. The algorithm assigns scores to attributes based on their ability to distinguish between classes among otherwise similar instances. The results of reliefF on the stroke dataset are shown below:



The results of ReliefF are shown here:

After applying a cut-off value of **0.001**, the only selected attributes are 17, 12, 5, 30, and 10.

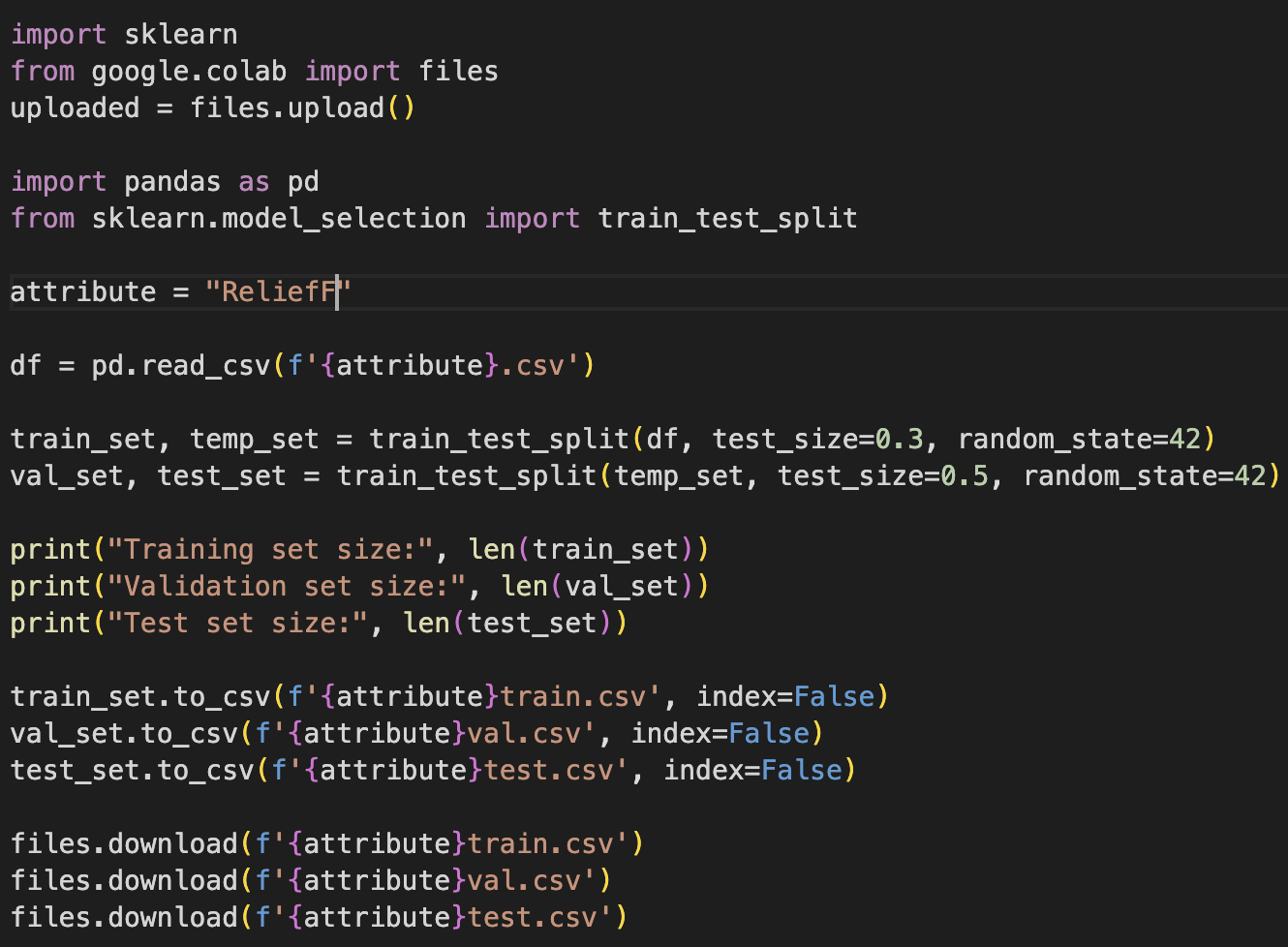
**Personal Selection:**

Attributes for Removal:

* **patient\_gender**: We don’t believe gender has a significant impact on the risk of stroke.
* **dietary\_habits**: While important for overall health, dietary habits do not directly influence stroke risk.
* **work\_type\_of\_patient**: A patient’s job type is too indirectly related to stroke risk; there are better, more direct features in the dataset such as cholesterol and blood pressure.
* **metabolic\_equivalent\_of\_task\_score**: Indicates physical activity level via metabolic expenditure rate; again, too indirectly correlated with stroke risk.
* **marital\_status**: Demographic factor, not indicative of stroke risk.
* **alcohol\_intake**: We believe that only extremely excessive consumption of alcohol would have an effect on stroke risk (“excessive drinker” was not a possible value, only “frequent drinker”)
* **residence\_type**: Location impacts lifestyle/healthcare, but not directly correlated to stroke risk, again.
* **All Symptoms** (“Blurred Vision,” “Seizures,” “Difficulty Speaking,” “Weakness,” “Confusion,” “Headache,” “Dizziness,” “Severe Fatigue,” “Loss of Balance,” and “Numbness”): All symptoms are self-reported, but we wanted to maintain objectivity when predicting stroke risk.

**Train/Test/Validation Split**

After selecting 5 different pools of attributes, we created 5 copies of the final.csv post preprocessing, from which we selected the different attribute pools. After this, we performed a 70%/15%/15% train/validation/test split on each new dataset using google colab and scikit-learn. 70% of the data will be used to train the model, 15% will be used to validate the model and potentially readjusting the hyperparameters or dataset, and 15% will be used to test the model. This split resulted in 7244 patients for training, 1552 patients for validation, and 1553 patients for testing.



**4.2 Classifier Models**

**RandomForest:**

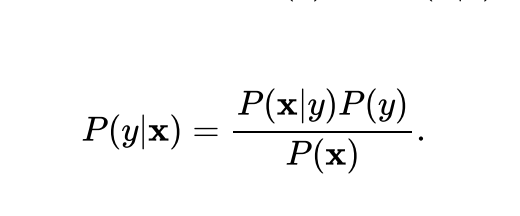
The Random Forest classifier builds a collection of decision trees (hence, called a “forest”), where each tree independently predicts the class for a given input. The final classification is determined by a majority vote from all trees, meaning the most common prediction among all trees is chosen. This approach helps reduce the risk of overfitting by taking the average of multiple predictions.

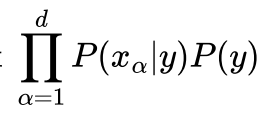
**J48:**

J48 is a machine learning algorithm that creates decision trees to classify data. It builds the trees by analyzing training data to find the most informative features, using techniques to handle missing data and to prune the tree to prevent overfitting. This ensures that the model remains generalizable to new input data.

**NaiveBayes:**

Naive Bayes is a probabilistic classifier that applies Bayes’ Theorem. It calculates the class probability using the formula:



In the context of the Naive Bayes classifier, X represents the set of features of the data (X = x1, x2, …xn), where each xi is an individual feature. Y represents the class variable the model is trying to predict (stroke vs. no stroke in this case). The above equation can be rewritten as:

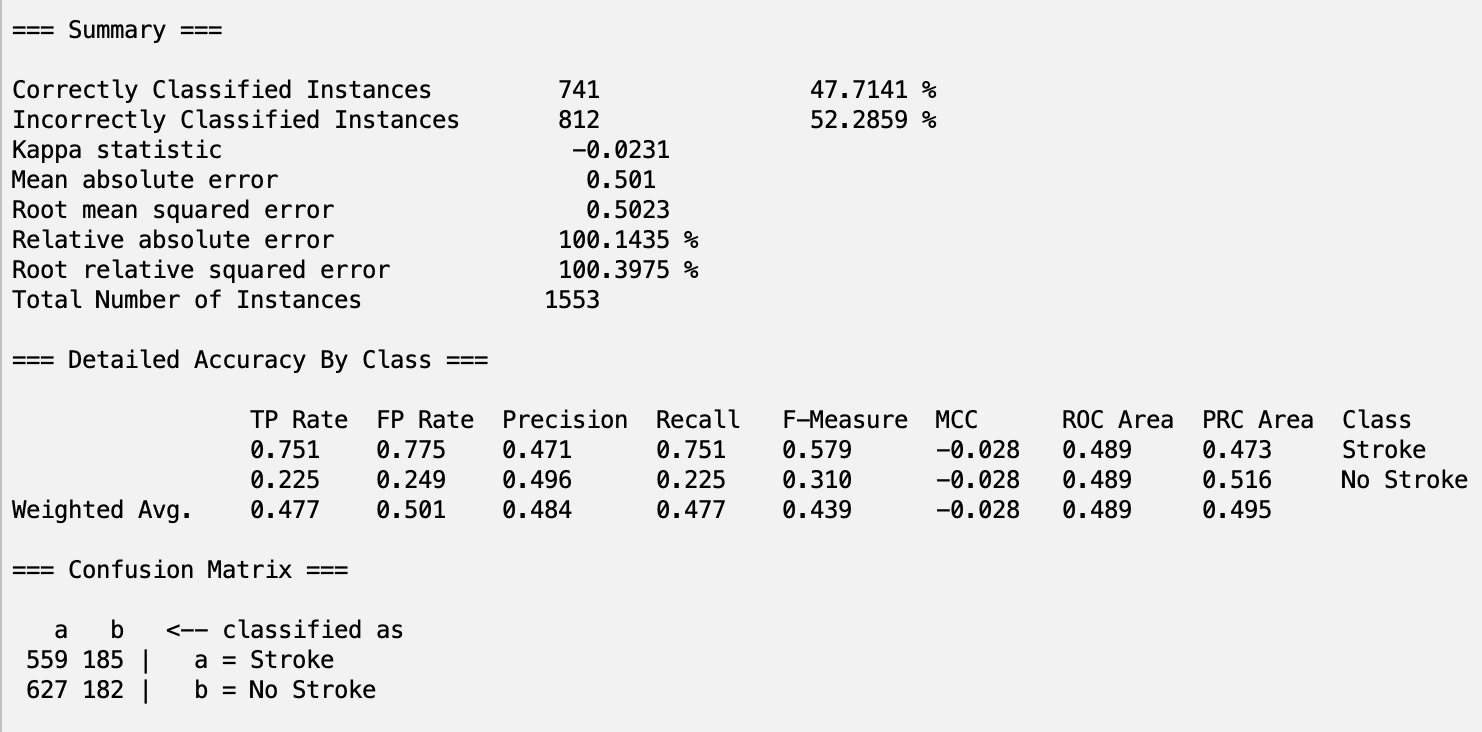
Naive Bayes uses these features X to estimate the probability of each possible class Y based on observed attributes.

**Decision Table:**

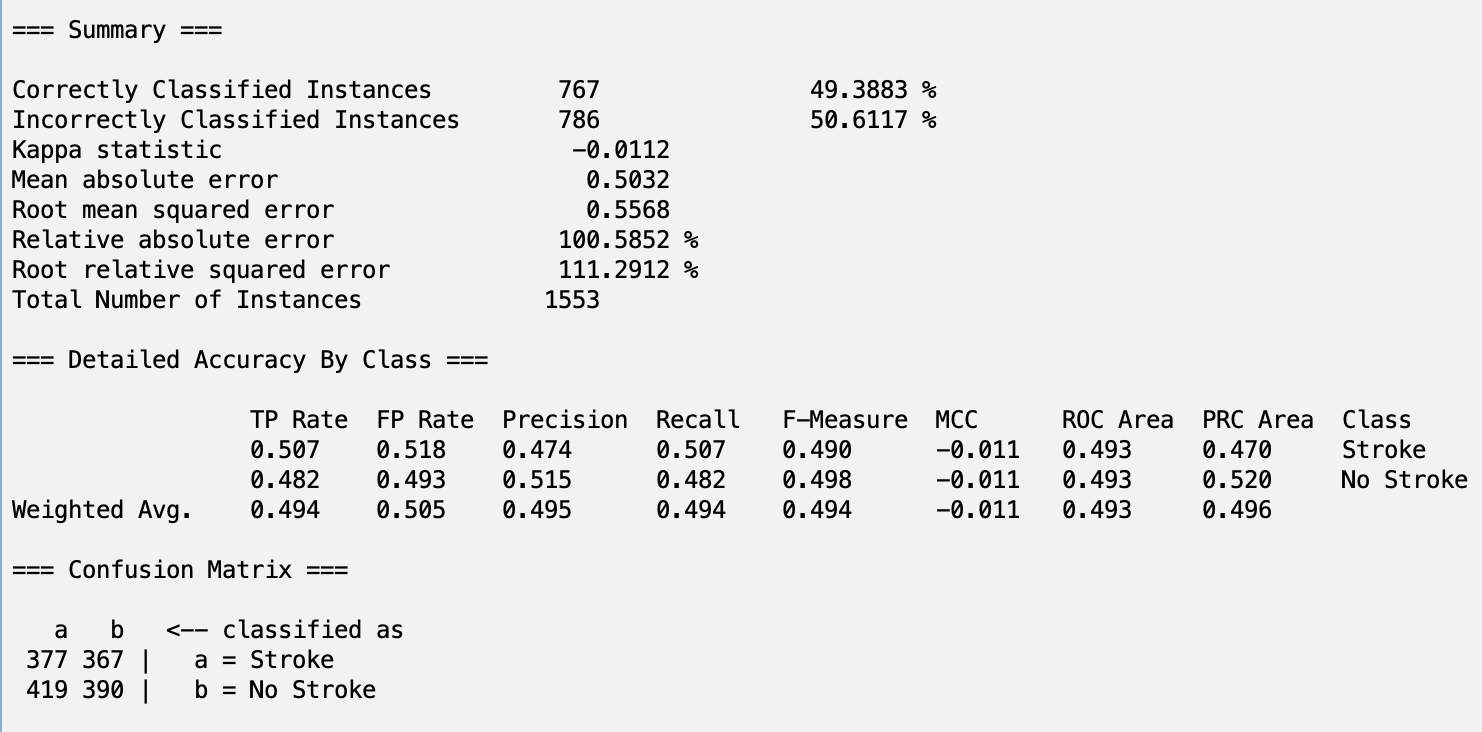
The DecisionTable classifier works by first creating a decision tree where each node tests an attribute and branches out based on the outcomes. It then translates these paths into a table of rules, each representing a potential decision path from the tree’s root to its leaves. The final classification for each data point is determined by the most frequently occurring outcome among these paths. This simplifies the classification process into an easily interpretable tabular format, making it a more straightforward classification algorithm.

**Part 5 – Results and Evaluation**

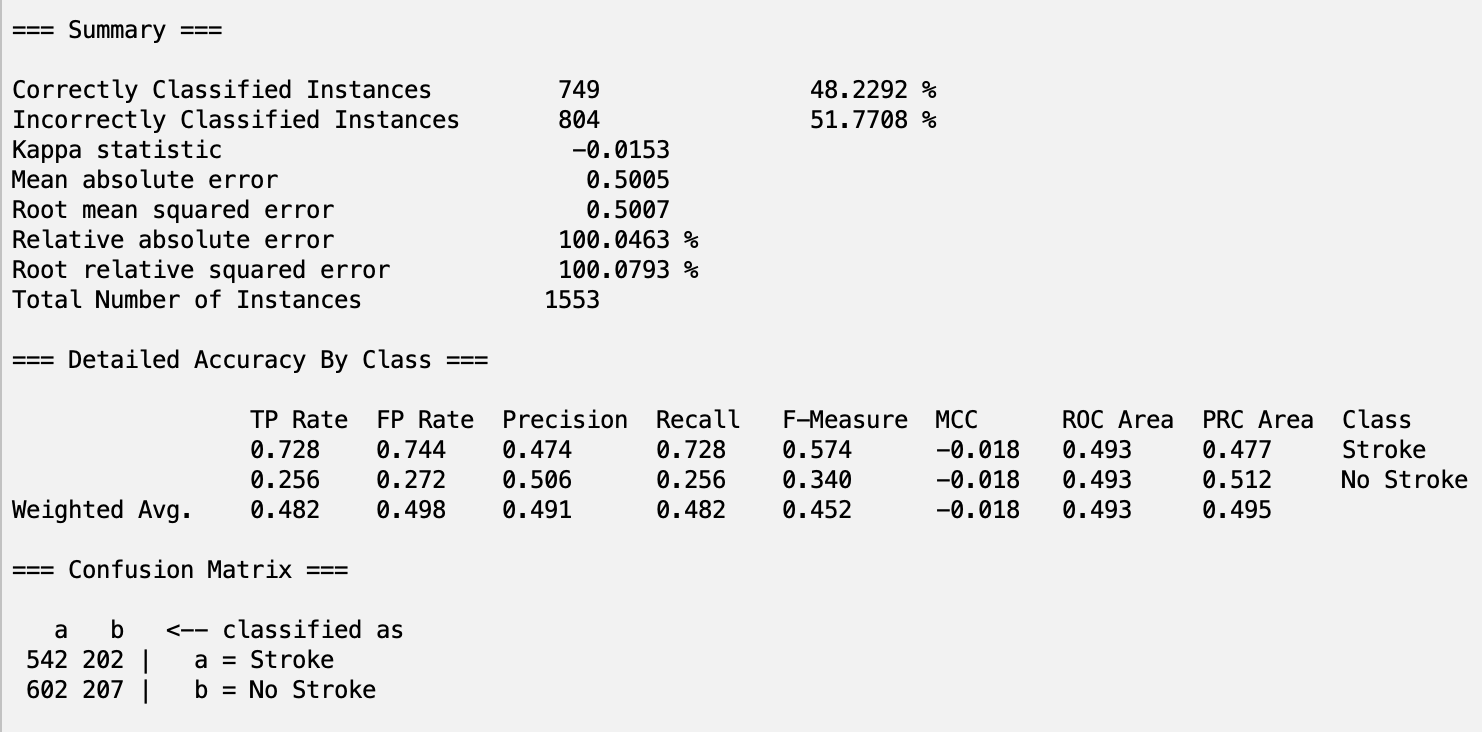
**5.1 Results**

**ReliefF with J48**

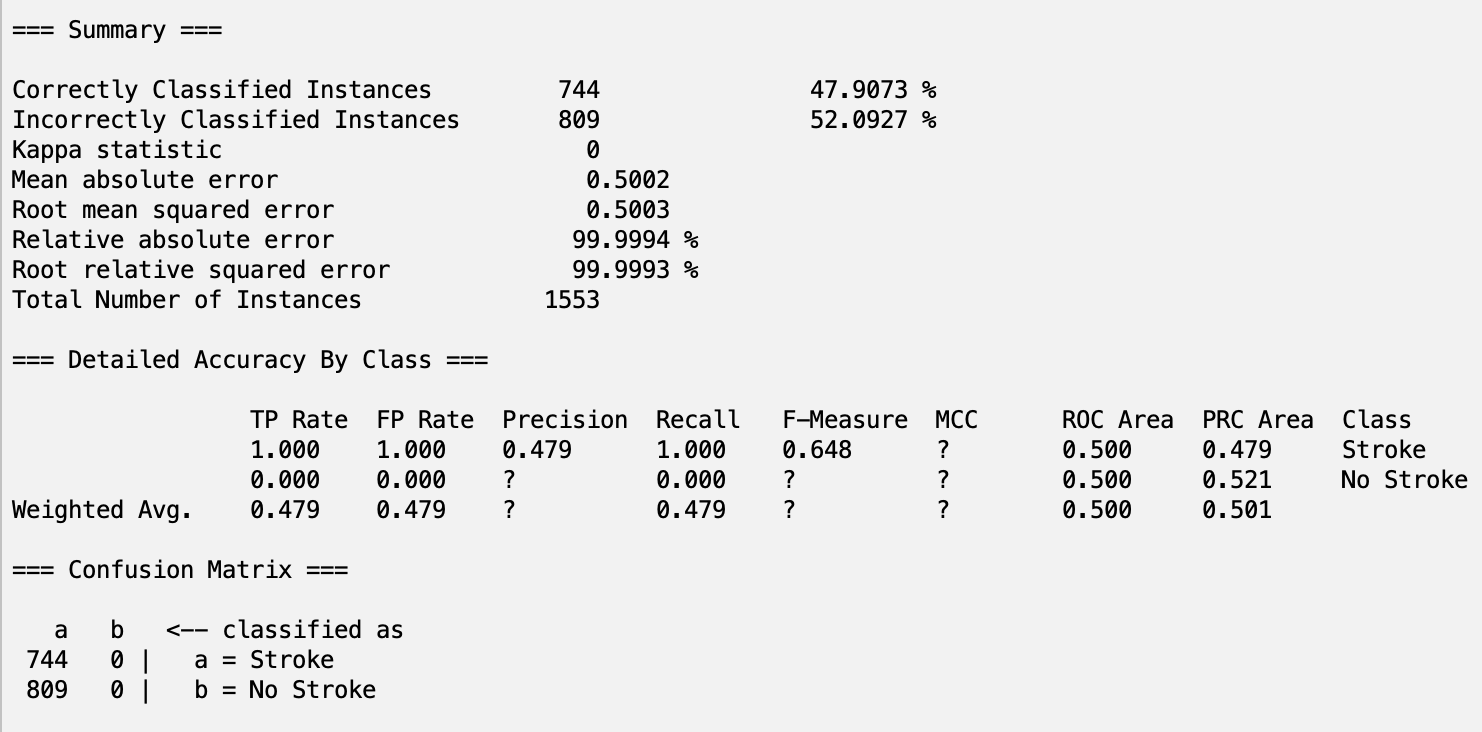
**ReliefF with RandomForest:**



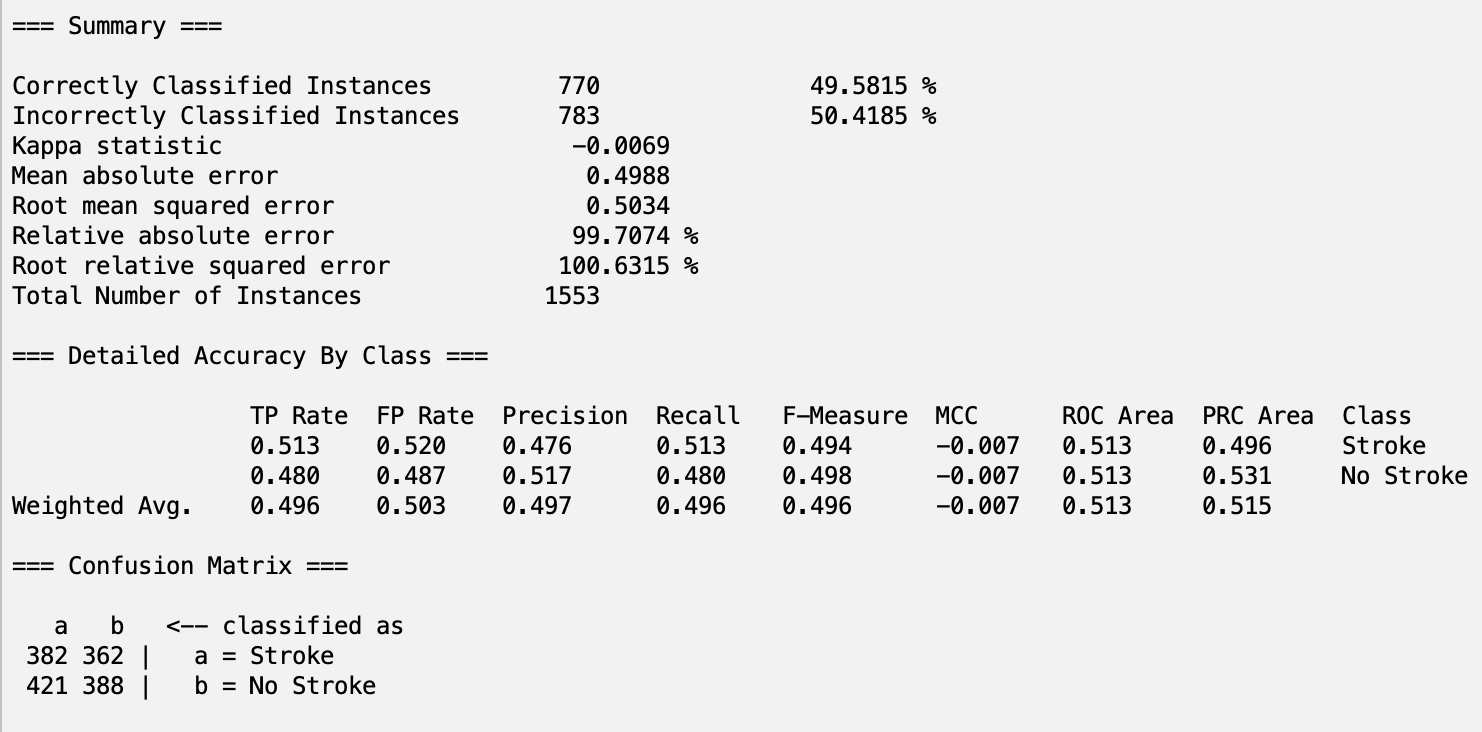
**ReliefF with NaiveBayes:**

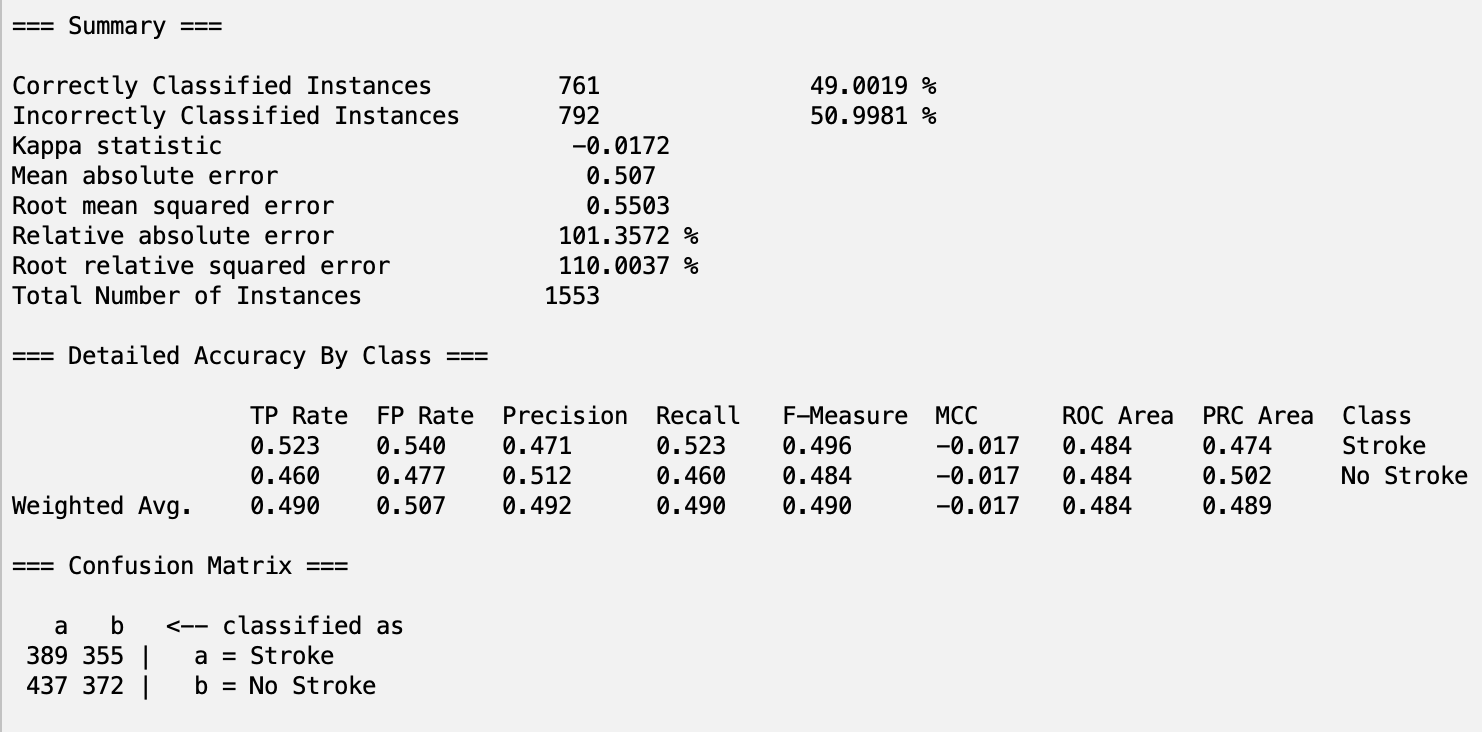
****

**ReliefF with Decision Table:**

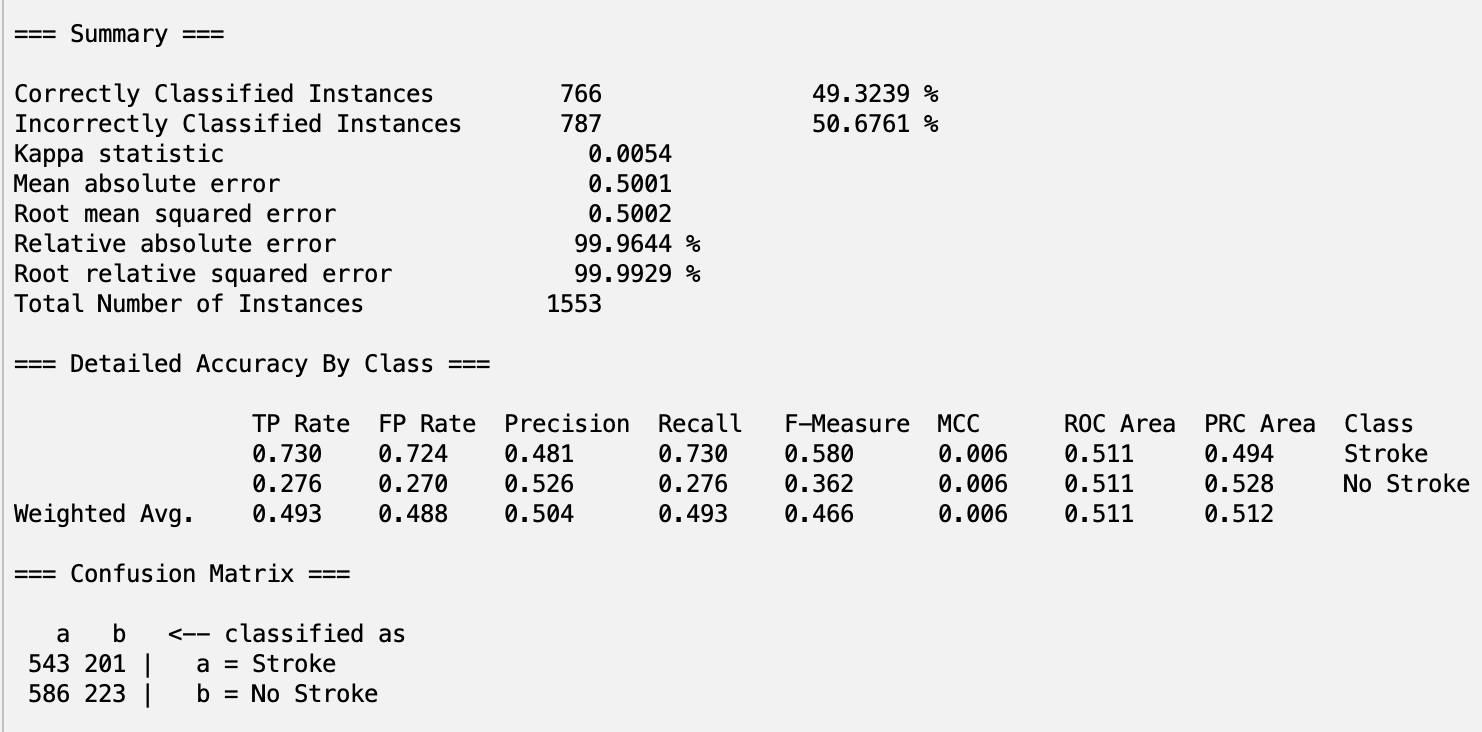
****

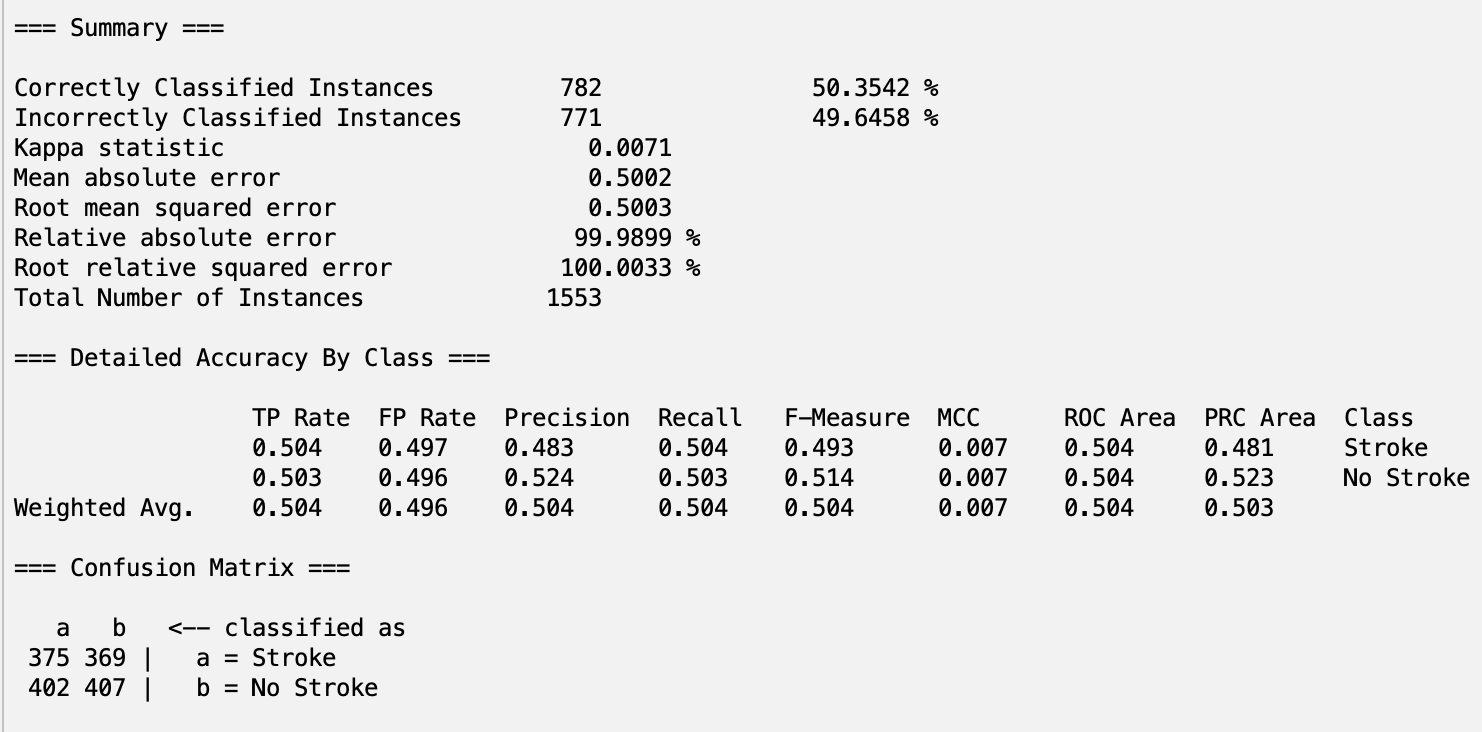
**OneR with J48:**

****

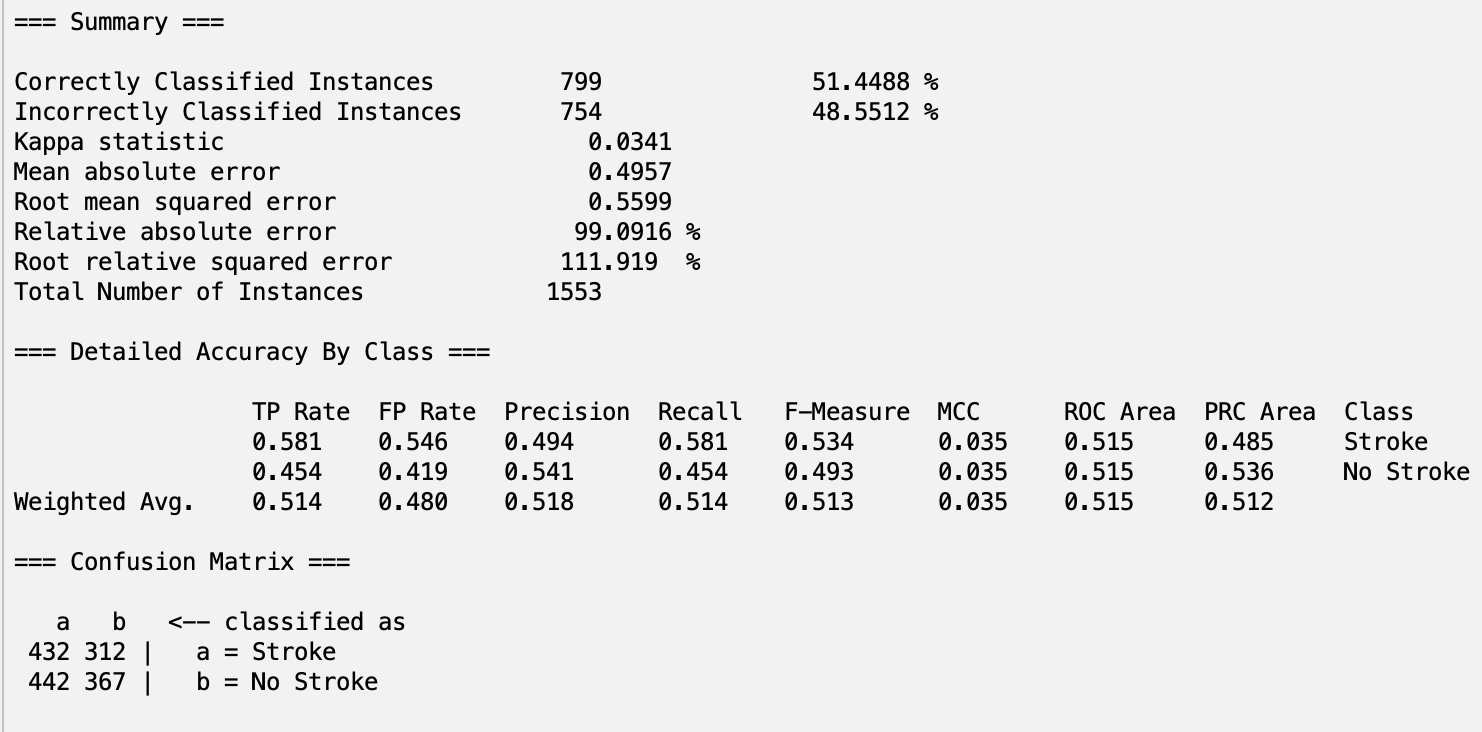
**OneR with RandomForest:**

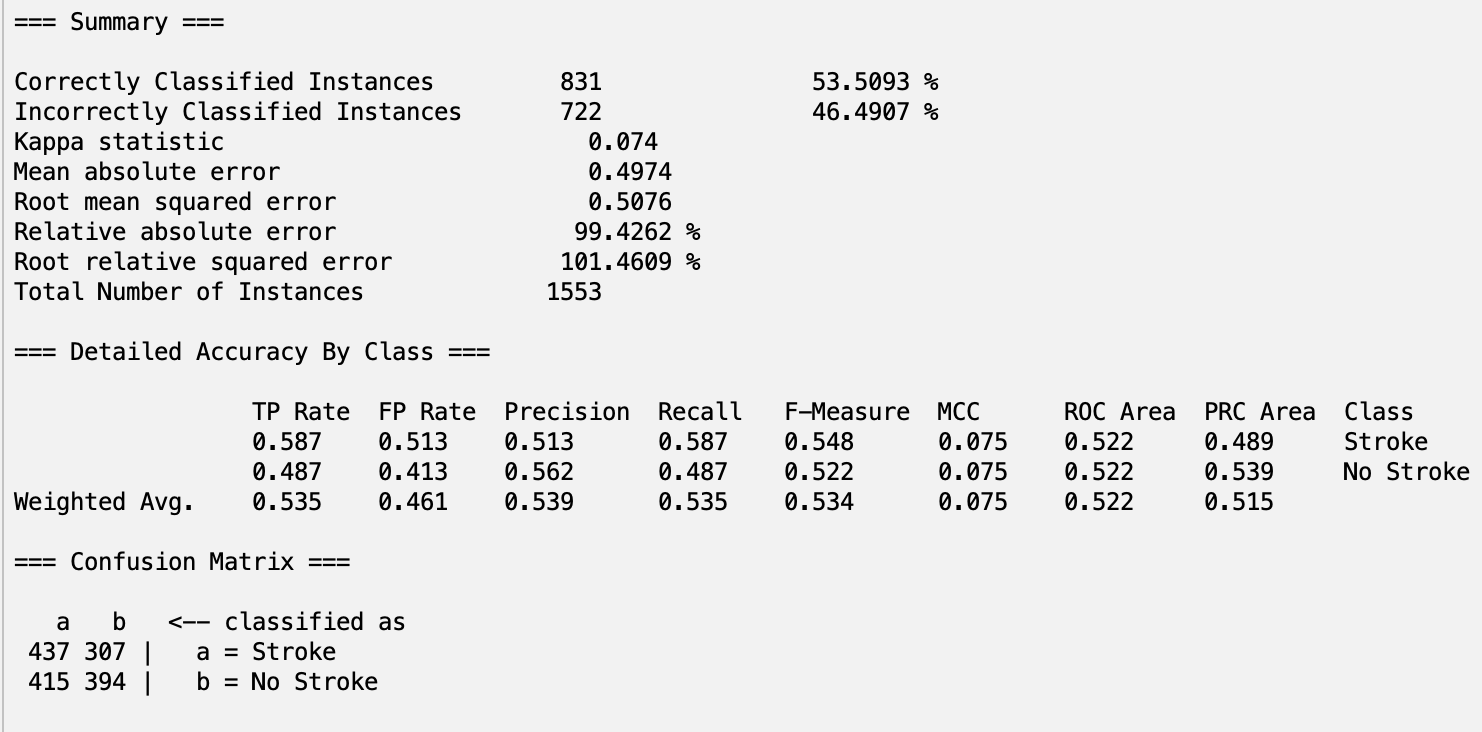
**OneR with NaiveBayes:**

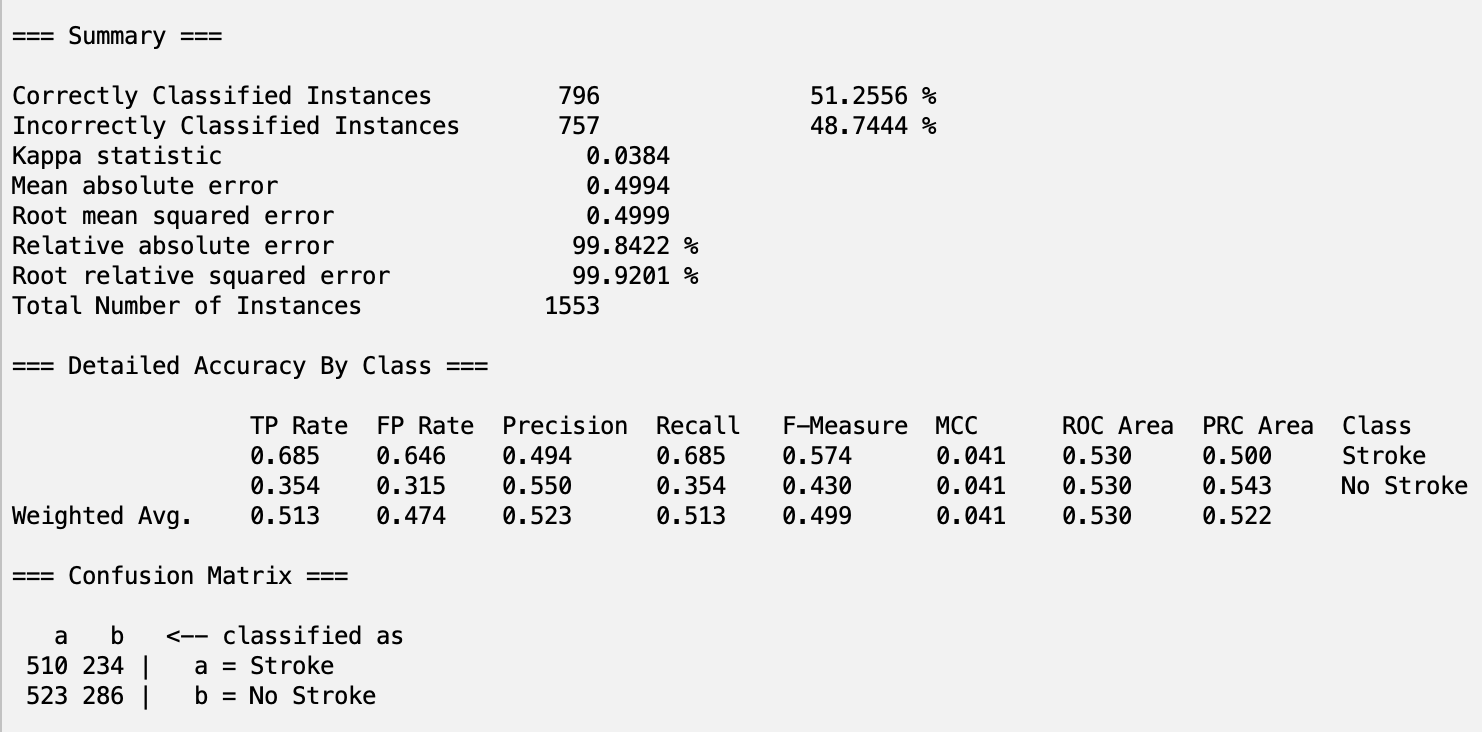
****

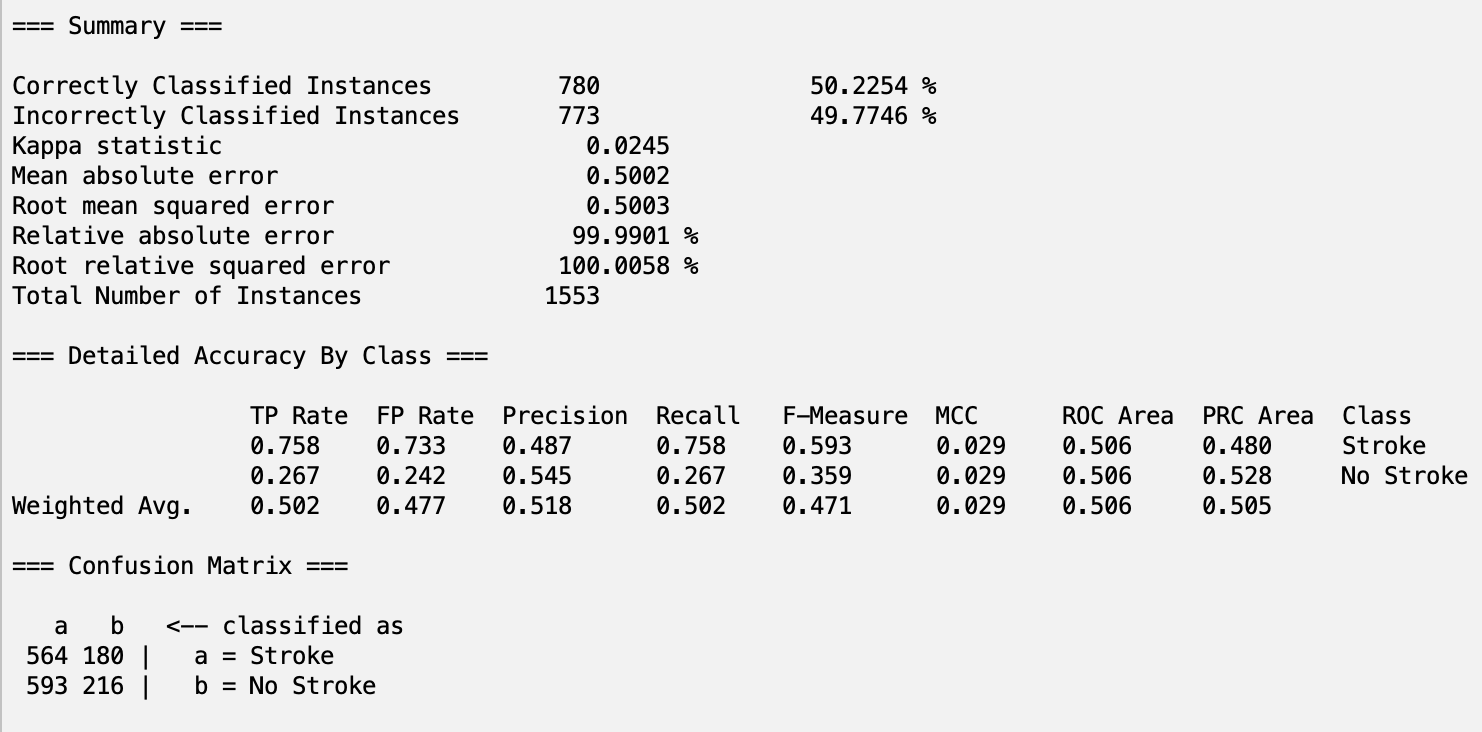
**OneR with Decision Table:  
**

**CorrelationAttributeEval with J48:**

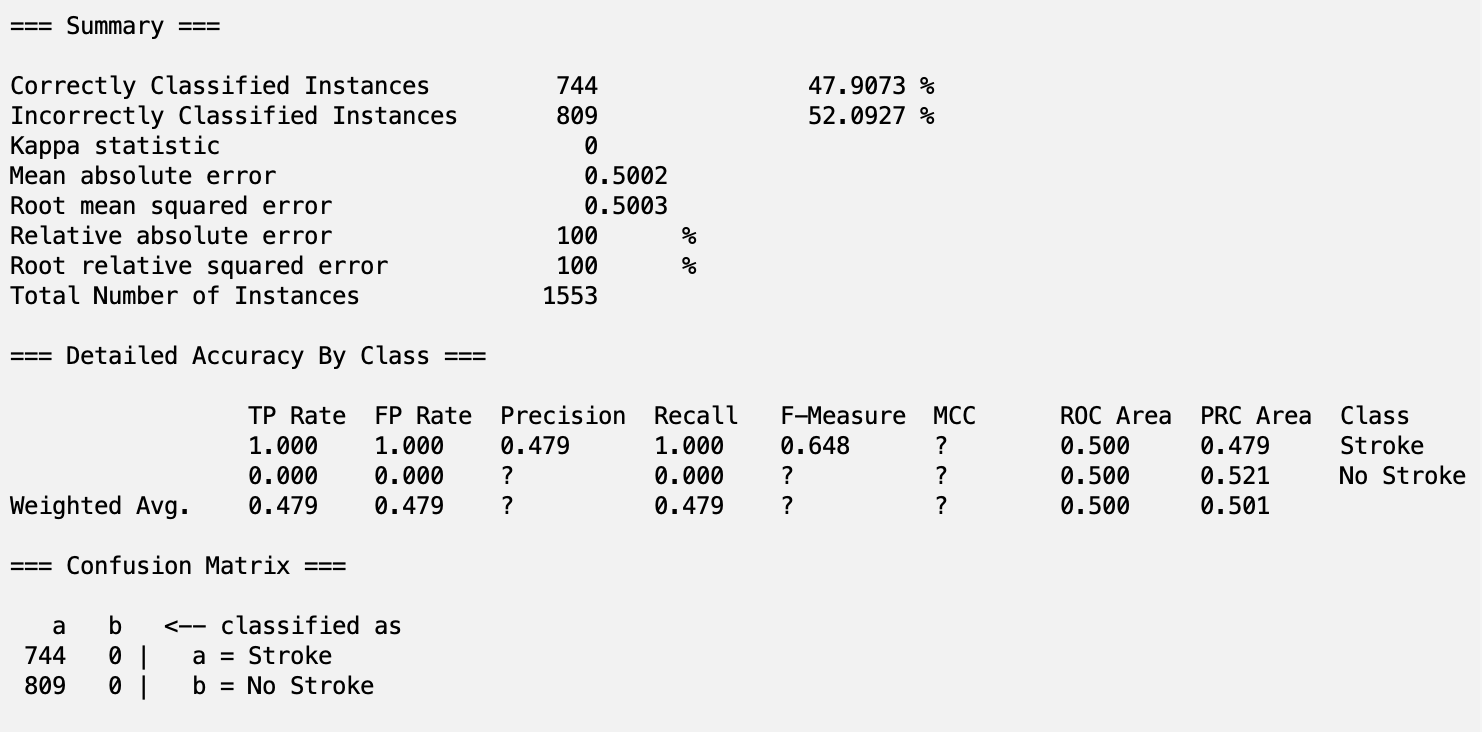


**CorrelationAttributeEval with RandomForest:**

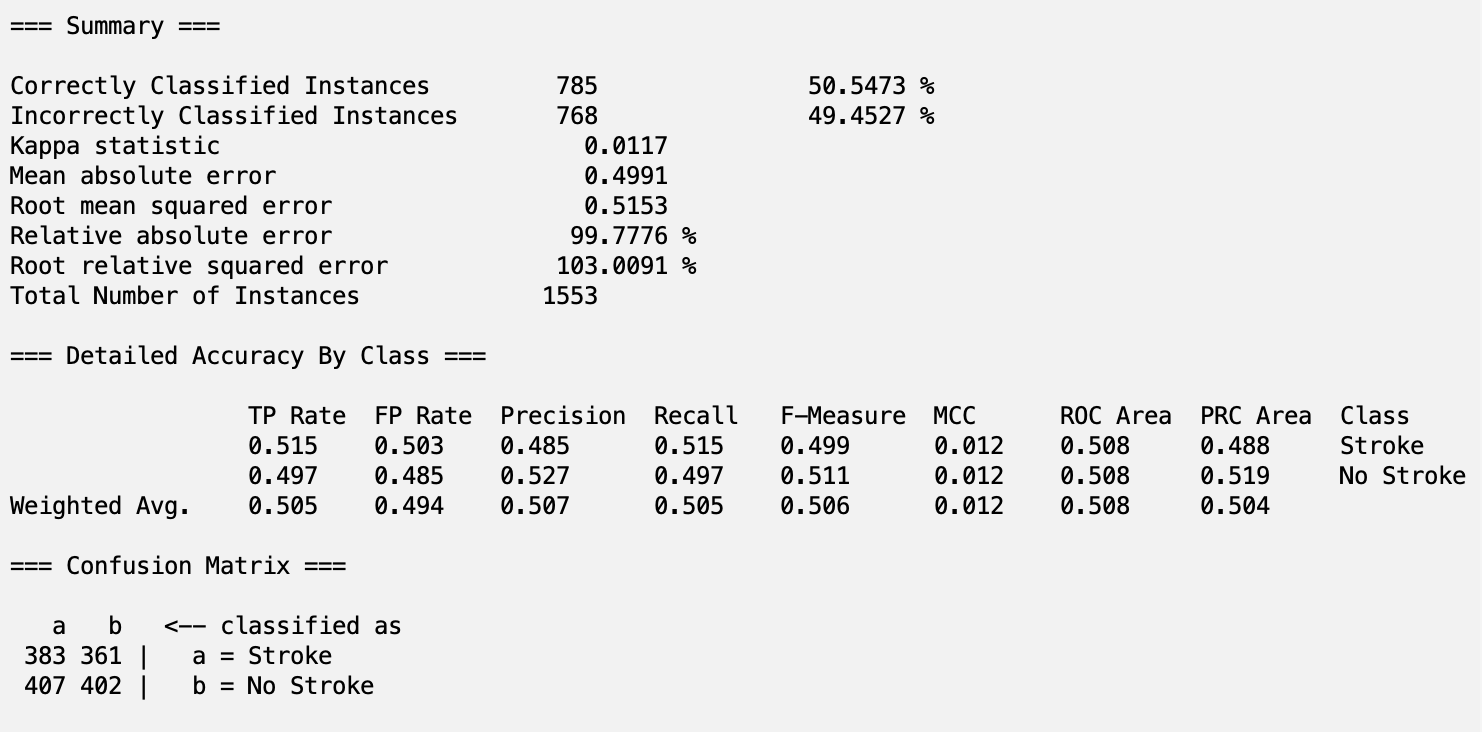
**CorrelationAttributeEval with NaiveBayes:**

**CorrelationAttributeEval with Decision Table:**

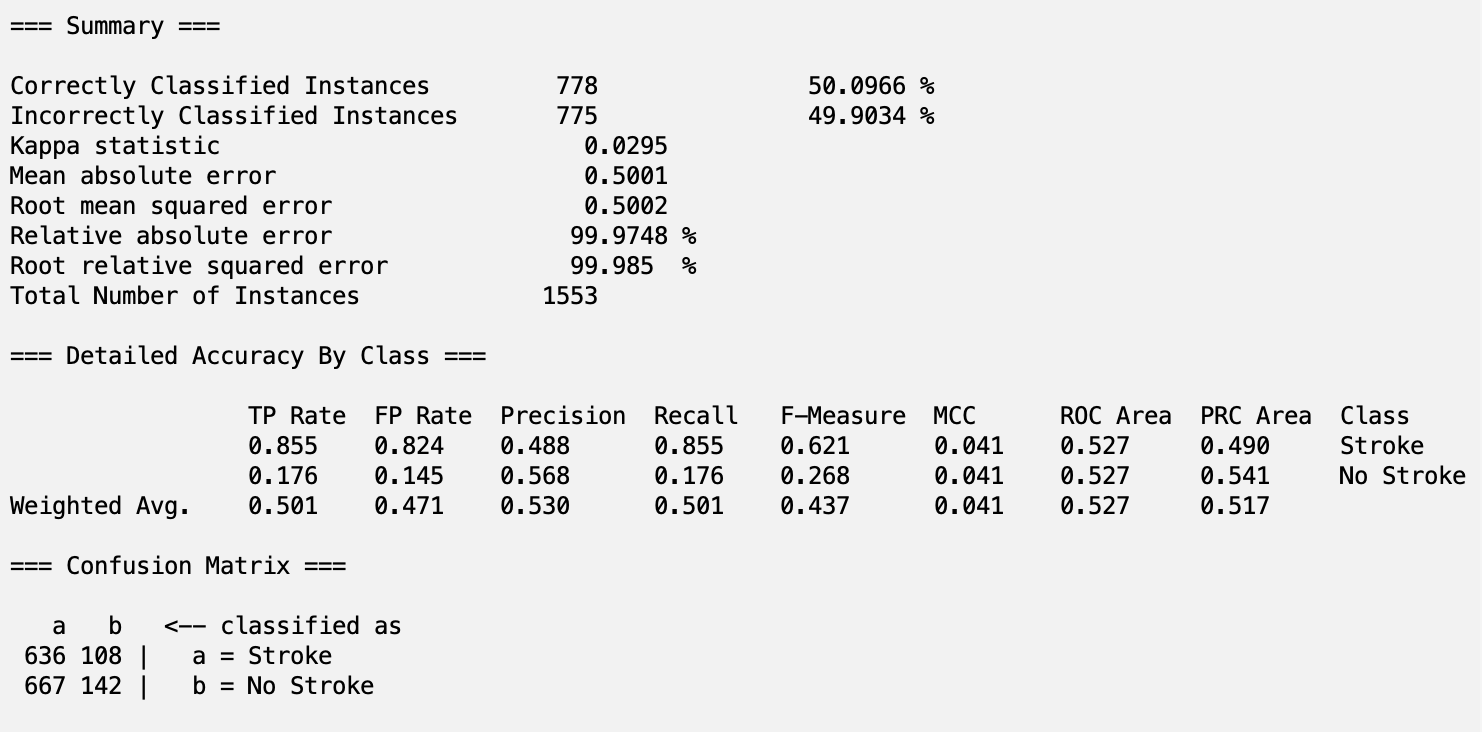
**PCA with J48:**



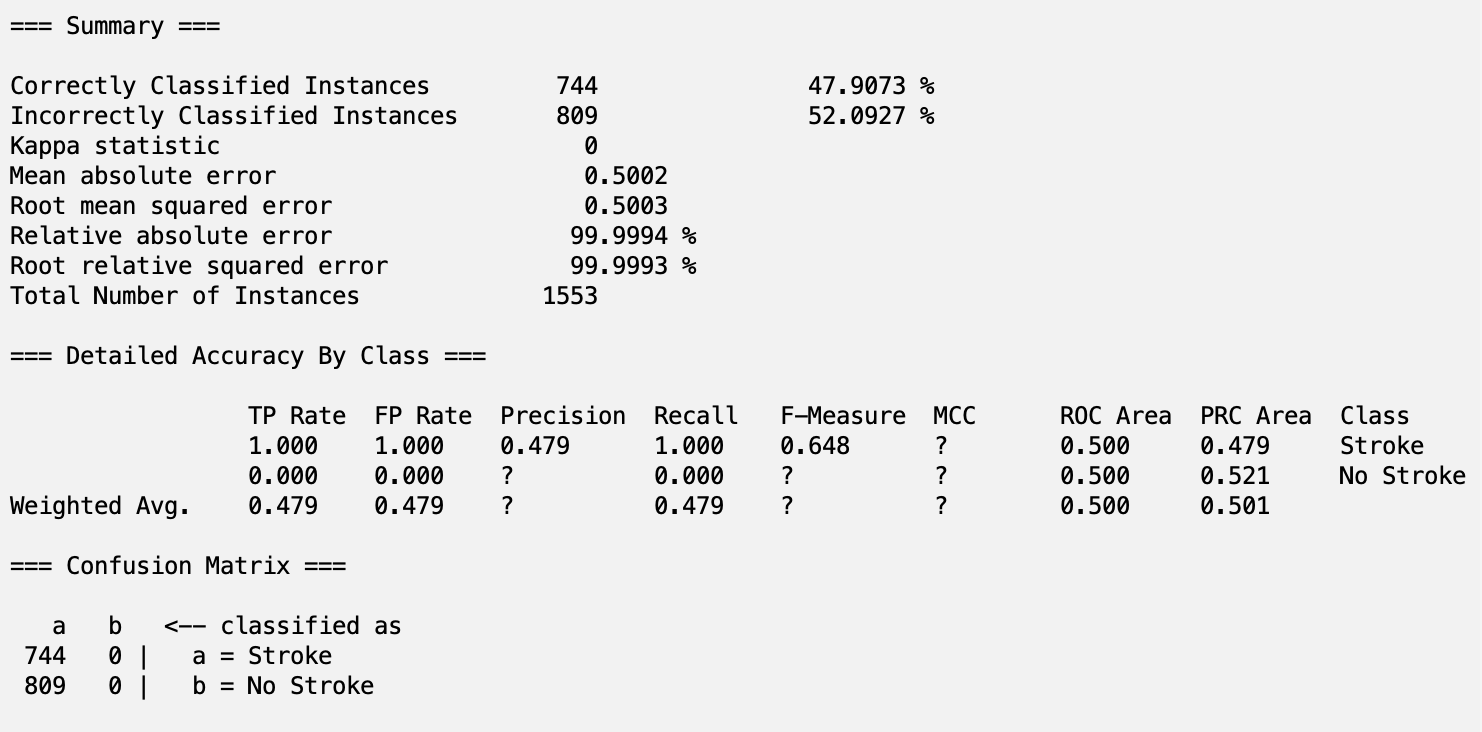
**PCA with RandomForest:**

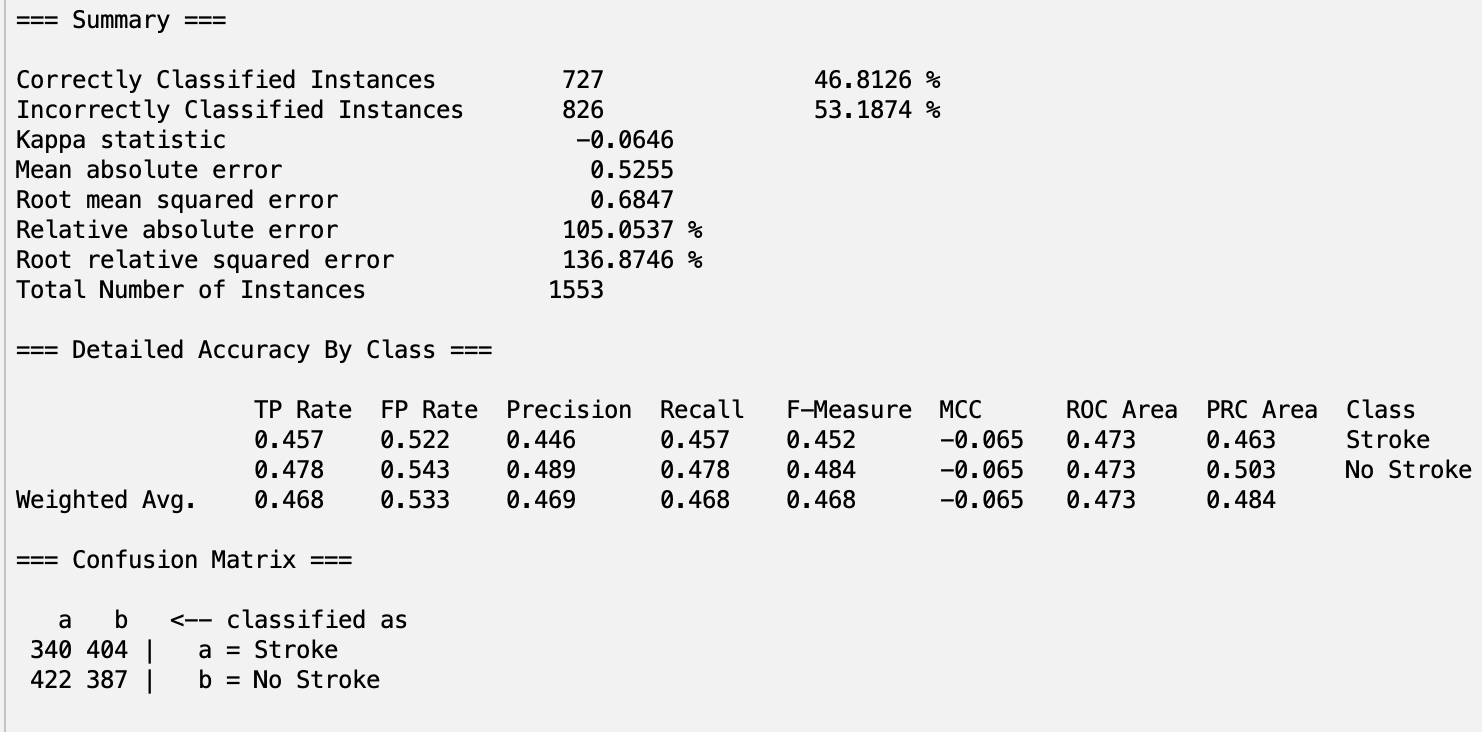


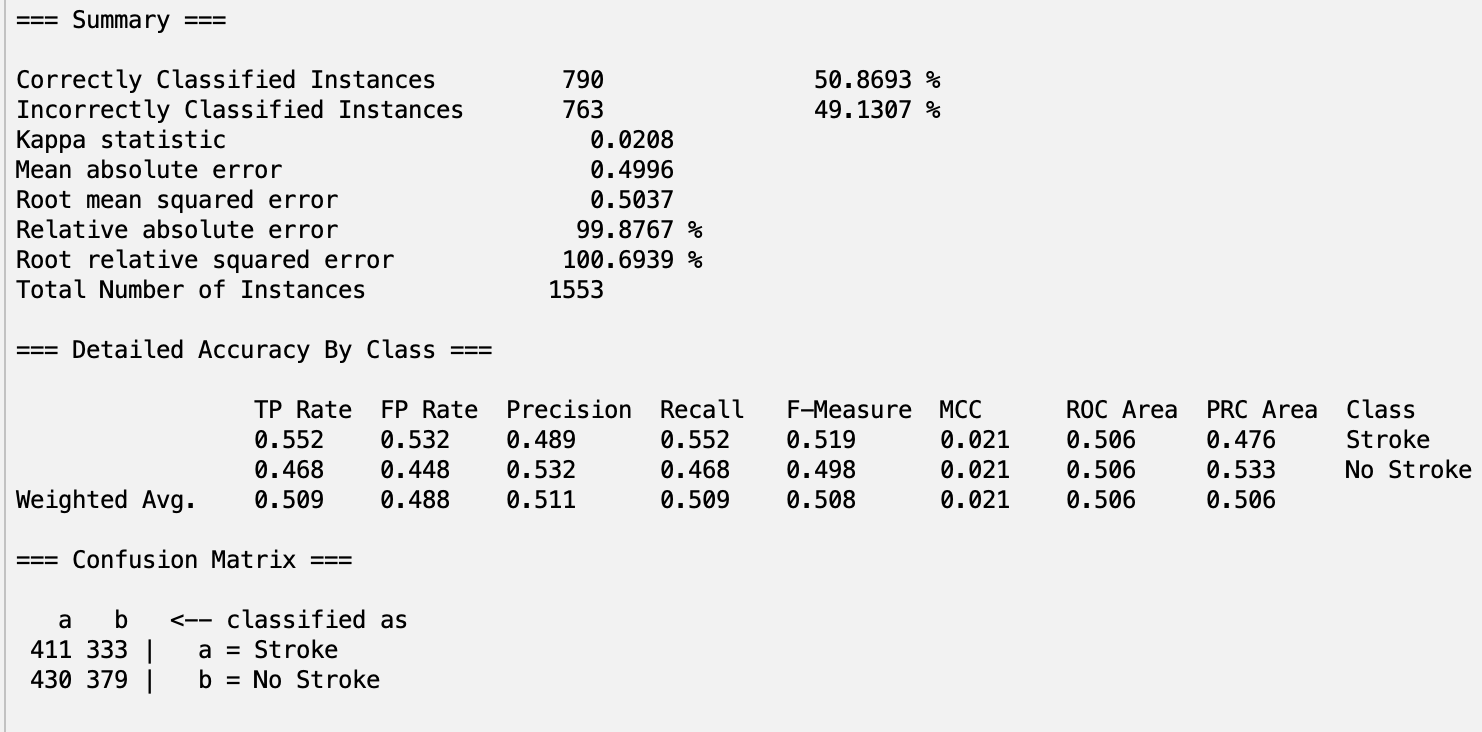
**PCA with NaiveBayes:**

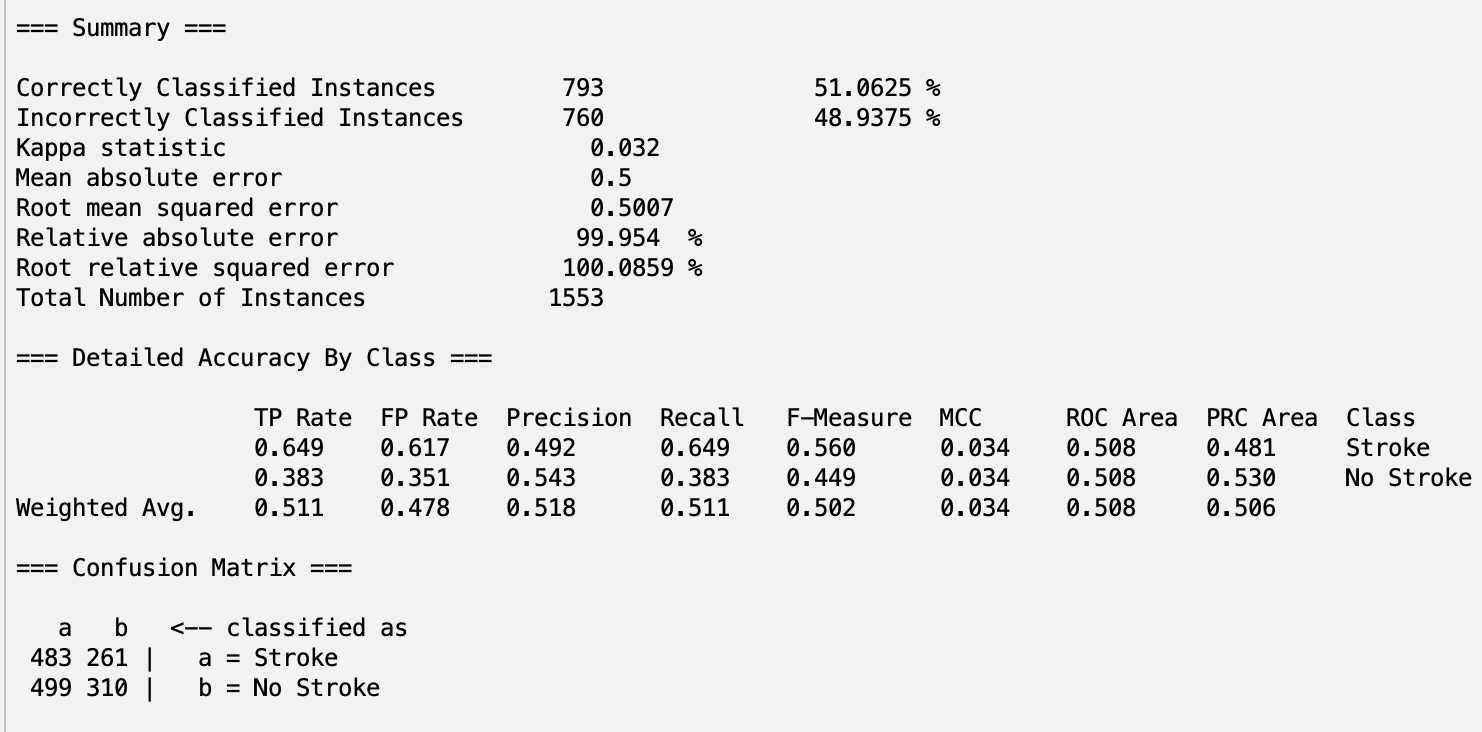


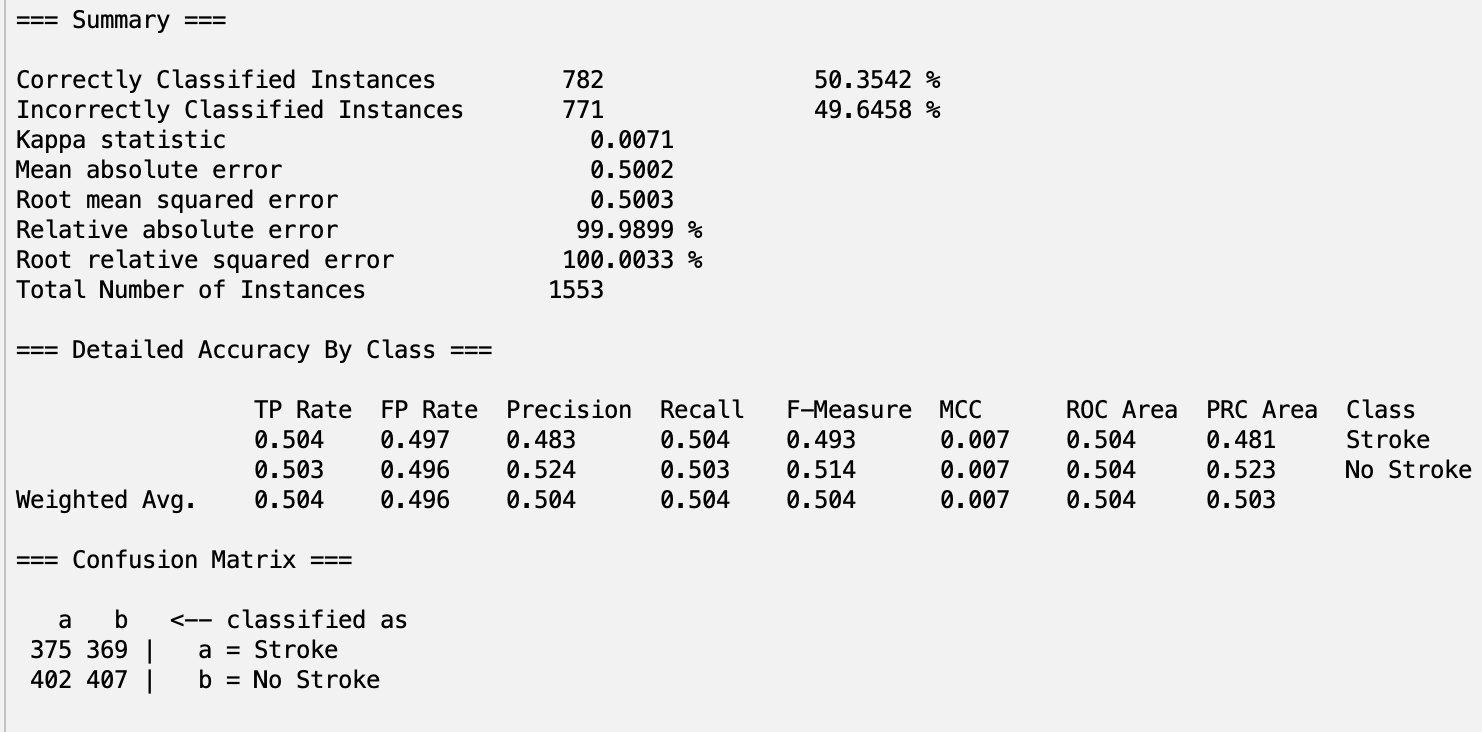
**PCA with Decision Table:**

****

**Personal Selection with J48:**

**Personal Selection with RandomForest:**

**Personal Selection with NaiveBayes:**

**Personal Selection with Decision Table:**

**5.2 Evaluation**

Through 4 different classifiers and 5 attribute selection methods, we created 20 different classification models to predict whether or not a patient is at risk of having a stroke based on a variety of factors. The 5 classifier models with the highest accuracies are listed below along with TP rate, FP Rate, and ROC Area, respectively:

1. CorrelationAttributeEval with RandomForest – 53.51%, 0.535, 0.461, 0.522

2. CorrelationAttributeEval with J48 – 51.45%, 0.514, 0.480, 0.515

3. CorrelationAttributeEval with NaiveBayes – 51.26%, 0.513, 0.474, 0.530

4. Personal Selection with NaiveBayes – 51.10%, 0.511, 0.478, 0.508

5. Personal Selection with RandomForest – 50.87%, 0.509, 0.488, 0.506

Since **CorrelationAttributeEval with RandomForest** scored the highest accuracy as well the highest TP rate and lowest FP rate of the top 5, we believe that this classifier model is the best model out of the 20 for predicting the possibility of a stroke for a patient. However, it is important that we acknowledge the somewhat poor results of our classifier models, with the highest accuracy only being slightly above 50%. We believe that these negative results were ultimately caused by the complexity of the problem at hand. First, the dataset contains many subjective attributes such as “Stress\_Level” and “Physical\_Activity.” As these are self-reported by the patients, they may contain inaccuracies, affecting the training process. Another potential reason is the nature of the many “Symptoms” attributes, as these symptoms may appear in various conditions, and these overlapping patterns may confuse the model. The final potential reason is the possibility of multicollinearity in our problem. Since strokes are a major health issue with various different causes, there is a high chance that many of the attributes are correlated amongst each other in large combinations. This complicates the problem, making it difficult to reduce attributes without losing valuable data, which in turn can lead to overfitting due to high dimensionality. We believe that our poor results were caused by a combination of these issues.

# **Part 6 – Conclusion and Reproducing Our Model**

In summary, the combination of CorrelationAttributeEval with the RandomForest classifier emerged as the most accurate model for predicting stroke risk in our study. However, the challenges of subjective data via self-reporting and potential multicollinearity in stroke-related attributes indicates that there are certainly ways to better the model. To increase accuracy in the future, datasets void of largely subjective data can be prioritized. Additionally, attribute selection algorithms capable of handling complex attribute selection (like multicollinearity) must be explored further to best improve accuracy. By improving on these aspects, we strive to develop a more precise machine learning tool with higher accuracy.

**Steps to Reproduce Our Model: CorrelationAttributeEval with RandomForest →**

1. Open Weka and load the **final.csv** achieved after following the preprocessing steps

2. Navigate to the **Select attributes** tab. Under Attribute Evaluator click Choose > attributeSelection > **CorrelationAttributeEval**.

3. A popup will appear informing you that you must use the **Ranker** search method in order to perform CorrelationAttributeEval. Click yes. If the popup does not appear, then the correct search method is already selected.

4. Click on the dropdown menu labeled “No class,” and select “**(Nom) Diagnosis**”

5. Click start, then **take note** of the ranked attributes with a value **> 0.01**

6. Return to the Preprocess tab, and check the box to the left of all attributes that you did not take note of in **Step 6** (Excluding Diagnosis). Press the remove button.

7. Hit the **Save…** button, and export as **CorrelationAttributeEval.csv**.

8. Change the “attribute” variable to “CorrelationAttributeEval” in the Train/Test/Validation split code located in **Step 4.1** in Google Colab. Hit the run on the code block, then select the newly exported csv file from the file selector.

9. Return to Weka and load the **CorrelationAttributeEvaltrain.csv** that downloaded to your computer from the Google Colab code.

10. Navigate to the **Classify.** Under Classifier click Choose > classifiers > trees > RandomForest.

11. Click **Supplied test set** under Test options and select **CorrelationAttributeEvaltest.csv**, downloaded from the python code.

12. Hit **Start**. If a popup appears informing you that the train and test set are not compatible, hit **yes**.

13. Model can be found in our directory: **Classification/CorrelationAttributeEvalWithRandomForest.model**

**Part 7 – Team Members and Tasks Performed**

**Finding the Data & Building Proposal:** Chetan Maviti and Kanishk Sivanandam

**Preprocessing Initial Attempt:**  Chetan Maviti and Kanishk Sivanandam

**Preprocessing & Project Update:**  Chetan Maviti and Kanishk Sivanandam

**Non-Weka Attribute Selection Algorithm:** Kanishk Sivanandam

**Attribute Selection Algorithms and Classifiers:** Chetan Maviti

**Results Output:** Chetan Maviti and Kanishk Sivanandam

**Results Analysis:** Chetan Maviti and Kanishk Sivanandam

**Building Final Report:** Chetan Maviti and Kanishk Sivanandam

# 

# **Part 8 – References**

Data: <https://data.world/rohit0308/stroke-prediction-23>

<https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CorrelationAttributeEval.html>

<https://www.saedsayad.com/oner.htm>

<https://rasbt.github.io/mlxtend/user_guide/classifier/OneRClassifier/>

[https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used](https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186#)

<https://builtin.com/data-science/step-step-explanation-principal-component-analysis>

<https://www.mathworks.com/help/stats/relieff.html>

<https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

[https://medium.com/nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell](https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e)

<https://weka.sourceforge.io/doc.dev/weka/classifiers/rules/DecisionTable.html>

<https://www.cs.cornell.edu/courses/cs4780/2024sp/lectures/pdfs/lecturenote05.pdf>

<https://towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0>