Stroke Classification: Predicting Stroke Risk with Machine Learning Model

Milestone: Final Report

Group XIII
Harish Padmanabhan
Ragul Narayanan Magesh
857-421-1918 (Tel of Student 1)
617-792-7068 (Tel of Student 2)

padmanabhan.h@northeastern.edu
magesh.ra@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: 

Signature of Student 2: 

Submission Date: 15 April 2025

Problem Setting:

A Cerebrovascular accident (CVA) or commonly known as Stroke is a medical emergency caused by lack of blood flow to part of your brain. It is ranked as No.5 cause of death and a leading cause of disability in the United States. Every year, more than 795,000 people in the United States have a stroke. About 80% of strokes can be prevented or their impact minimized with early detection and intervention. The goal of this project is to create a machine learning classification model that can predict the likelihood of an individual experiencing a stroke based on their medical history, lifestyle and demographic factors.

Problem Definition:

The primary problem is to predict whether a person is likely to experience a stroke, based on a variety of factors such as age, hypertension, smoking status, and other health indicators. This is a binary classification problem, where the target variable is whether the individual has had a stroke.

Goals:

- Identify the key factors that most accurately predict stroke.
- Build a machine learning model that can predict stroke risk with high accuracy.
- Ensure that the model is effective and efficient with new data.
- Create a model that can be easily applied in clinical settings for early detection.

OKRs (Objectives and Key Results):

- **Objective 1:** Build a reliable accurate and effective predictive model for early detection and prevention of stroke risk
    - **Key Result 1:** Achieve a model accuracy of at least 85% to ensure reliable predictions
    - **Key Result 2:** Identify key predictive factors through feature selection
    - **Key Result 3:** Make sure that model performance performs well across different datasets
- **Objective 2:** Ensure the model is easily understandable for healthcare professionals
    - **Key Result 1:** Clearly visualize the key factors that contribute to stroke risk prediction

KPIs (Key Performance Indicators):

- **Accuracy:** To measure the overall correctness of the predictions
- **Precision:** To measure the ratio between true stroke predictions among all stroke predictions
- **Recall:** To measure the correctly identified stroke cases
- **F1 Score:** To achieve a F1 score of 0.80 and above
- **Prediction Time:** To create a model that takes less time to process new data and provide prediction

Data Source:

The data for this project is sourced from Kaggle named Stroke Prediction Dataset, that provides relevant health and demographic information. The dataset includes information from patients, such as medical history, lifestyle choices, and demographic data, which are used to predict stroke occurrence.

Data Source: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/data

Data Description:

The dataset contains both categorical and numerical features that represent the patient's demographic, health, and lifestyle information. The dataset also includes a target variable indicating whether the individual has experienced a stroke. The dataset contains 5110 records available with 12 columns. Below displayed are all 12 columns in the dataset:

Column	**Description**
Id	Identifier for individual patients [Numerical]
Gender	Gender of the patient [Categorical]
Age	Age of the patient [Numerical]
Hypertension	Patients have hypertension or not [Categorical]
Heart Disease	Patients have heart disease or not [Categorical]
Ever Married	Marital status of the patient [Categorical]
Work Type	Occupation status of the patient [Categorical]
Residence Type	Residence type of the patient [Categorical]

Avg Glucose Level	Average glucose level in patient's blood [Numerical]
BMI	Body mass index of the patient [Numerical]
Smoking Status	Patient smokes or not [Categorical]
Stroke	Patient affected by stroke or not [Categorical]

Data Pre-Processing:

Before training the models, several preprocessing steps were applied to prepare the dataset for analysis. Missing values in the BMI column (3.93% of entries) were imputed using the mean value, while ambiguous entries such as "Unknown" in the smoking status field were replaced with the most common category observed in the dataset. Irrelevant columns like the unique patient ID were removed to focus on meaningful predictors.

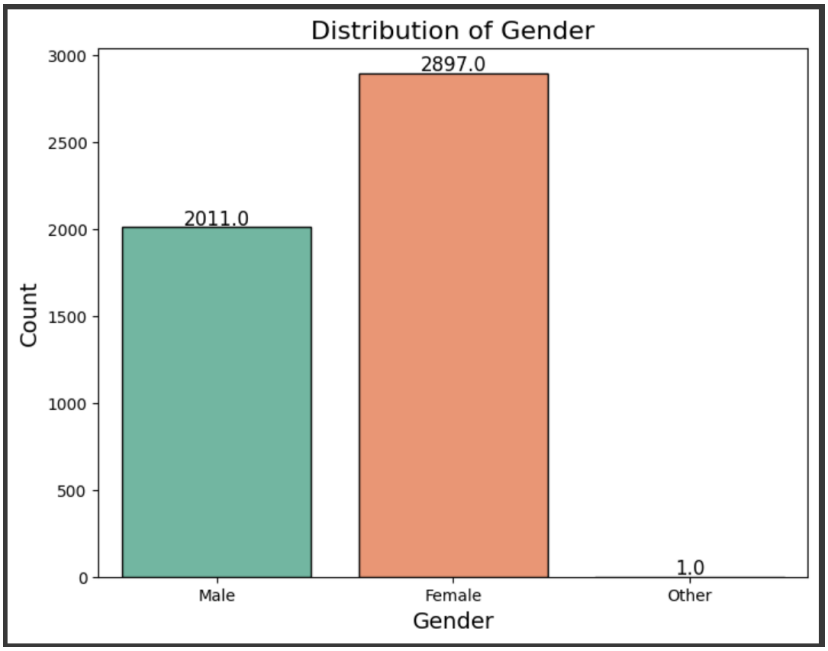
Categorical variables were encoded to numerical formats. Binary variables such as marital status and residence type were label-encoded, while multi-category variables like gender, work type, and smoking status were converted using one-hot encoding to preserve categorical distinctions.

A major challenge was the dataset's class imbalance, with stroke cases comprising a small fraction of the total. To mitigate bias toward the majority class, oversampling techniques were applied to the training data, which increased the representation of stroke cases and improved the models' ability to detect minority class patterns.

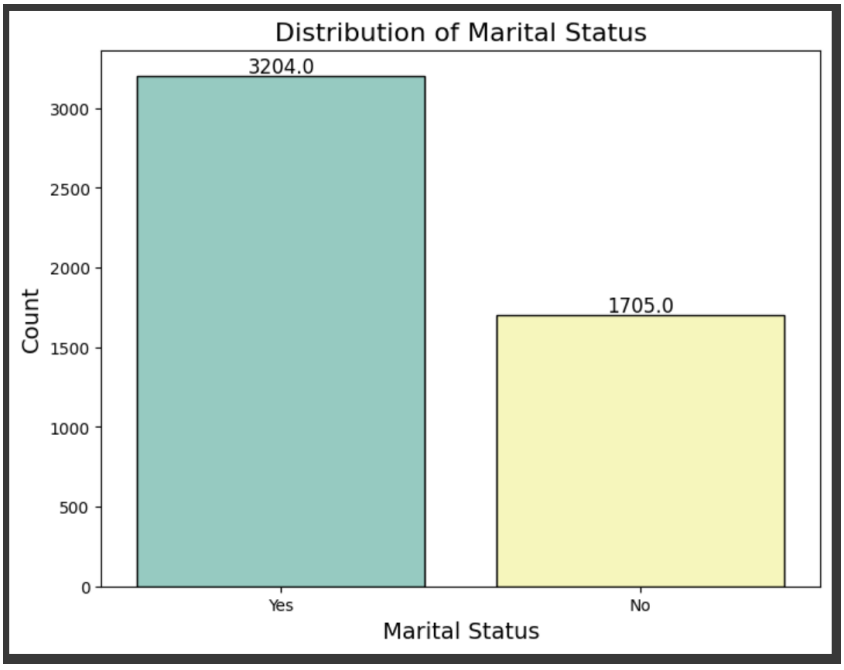
Lastly, numerical features such as age, average glucose level, and BMI were standardized to ensure consistency in scale, especially important for distance-based and gradient-based models. These preprocessing steps helped create a clean and balanced foundation for effective model training and evaluation.

Data Exploration:

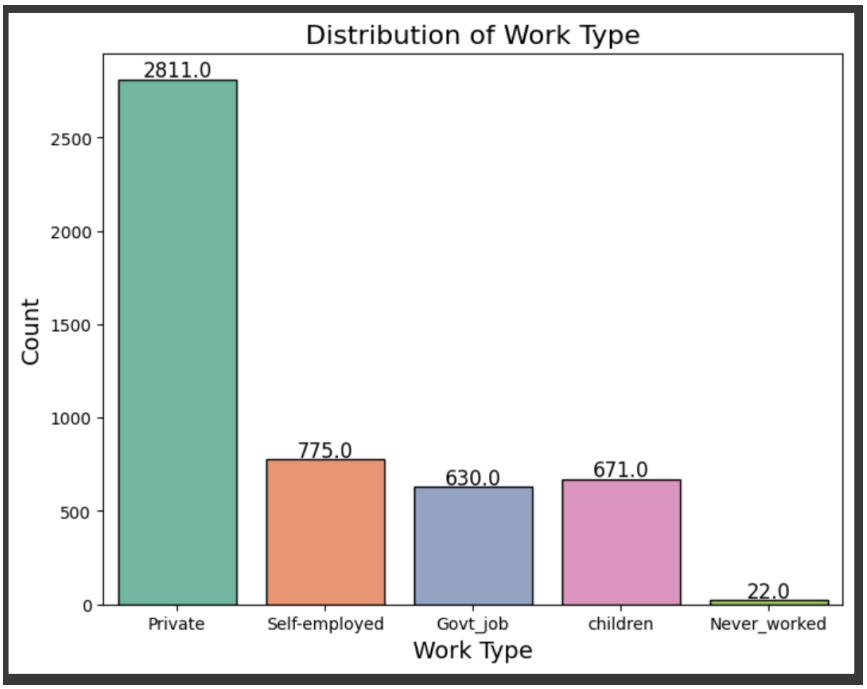
1. **Bar Charts**
   - Description of Gender:
   - The results suggest that a slightly higher representation of females in the dataset.



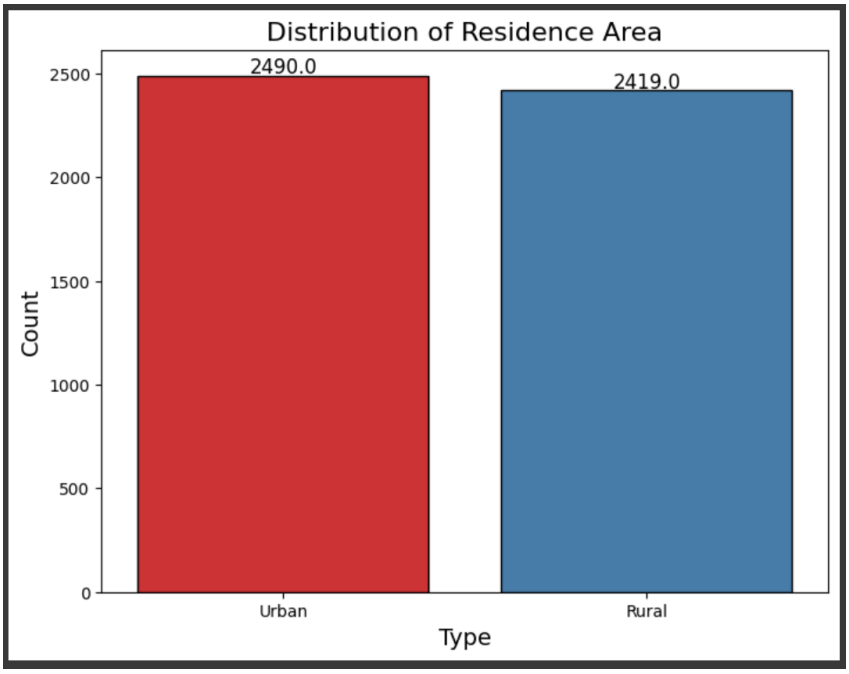
   - Distribution of Marital Status:
   - The results show most of the individuals in the dataset are married, i.e., 3204 individuals are married.



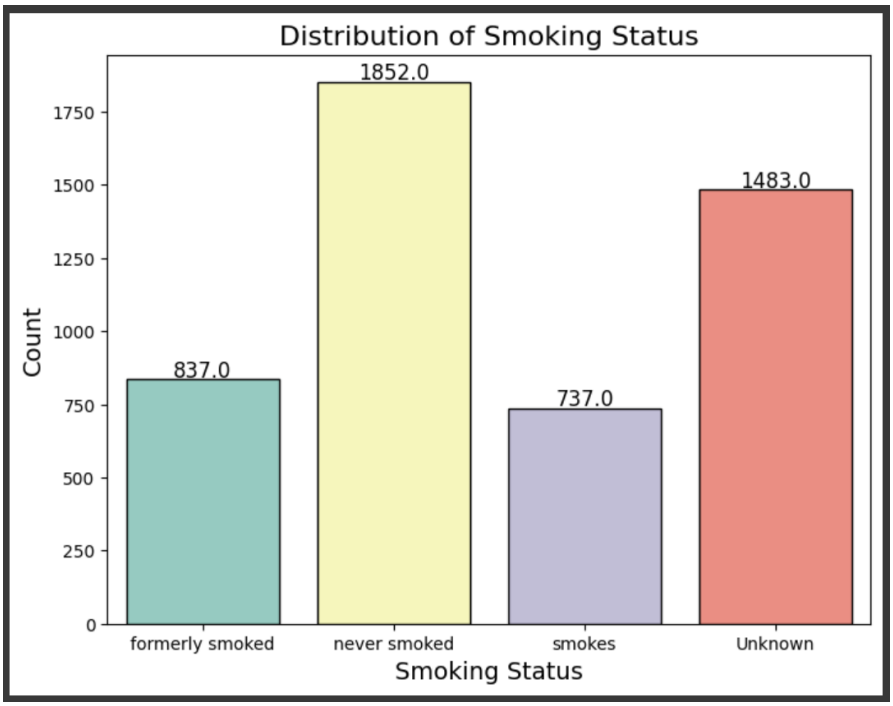
- Distribution of Work Type
- Results show that individuals who work in the private sector are the highest followed by self-employed, government jobs.
- Work type might be an indicator of stress level which could lead to a stroke.



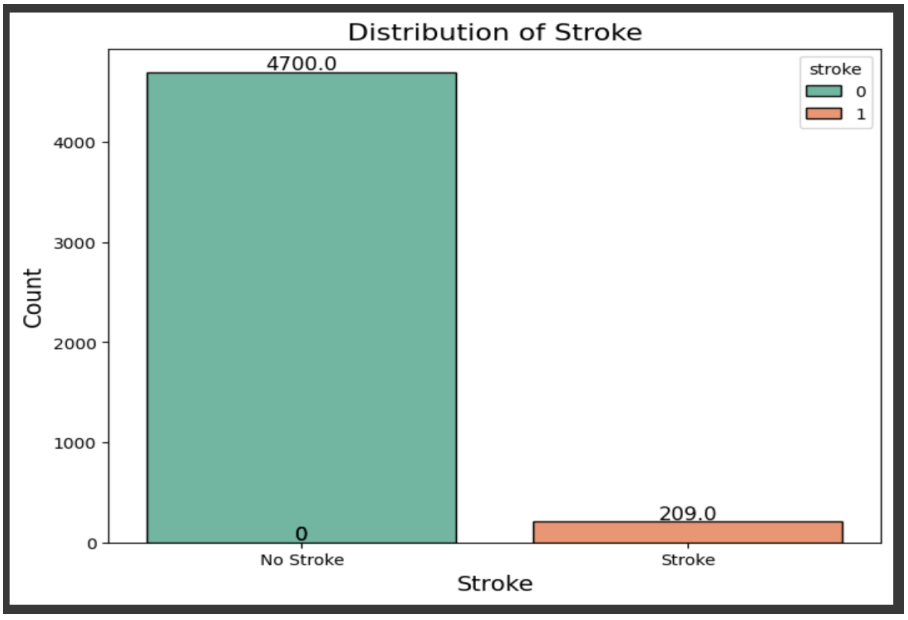
- Distribution of Residence Area
- The results show that the count of urban and rural residents is similar.
- Balance in data will contribute to analyzing stroke risk for both lifestyles.



- Distribution of Smoking Status
- 1852 individuals never smoked, 1483 individuals have unknown smoking status while 837 formerly smoked and 737 are current smokers.
- Smoking is a known stroke risk factor and further analysis will show its impact.



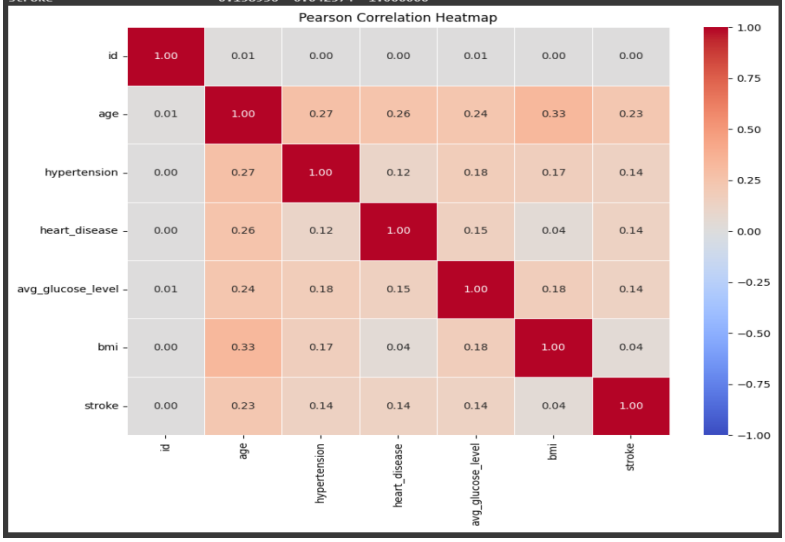
- Distribution of Stroke Cases
- The dataset contains 4700 individuals without stroke and 209 stroke cases.
- The dataset is highly imbalanced which requires techniques like oversampling to improve model performance.



2. Heat Map

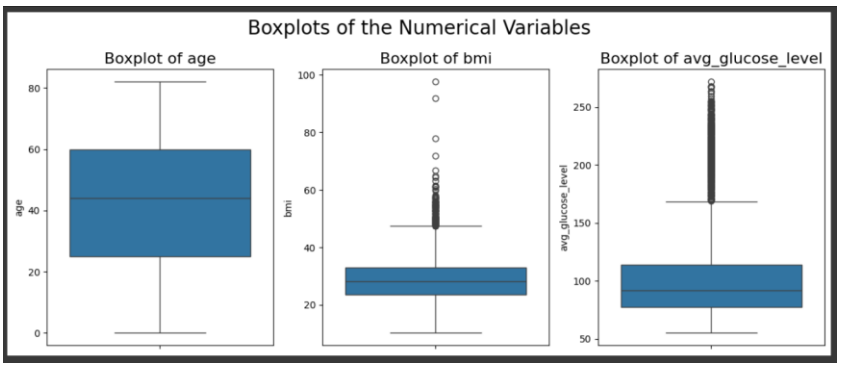
- Correlation Matrix
- The results show that age, hypertension, heart disease, and glucose levels show a moderate correlation with stroke whereas BMI has a very weak correlation with stroke.

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
id	1.000000	0.008984	0.001147	0.004016	0.006409	0.003084	0.004839
age	0.008984	1.000000	0.274425	0.257123	0.235838	0.333398	0.232331
hypertension	0.001147	0.274425	1.000000	0.115991	0.180543	0.167811	0.142515
heart_disease	0.004016	0.257123	0.115991	1.000000	0.154525	0.041357	0.137938
avg_glucose_level	0.006409	0.235838	0.180543	0.154525	1.000000	0.175502	0.138936
bmi	0.003084	0.333398	0.167811	0.041357	0.175502	1.000000	0.042374
stroke	0.004839	0.232331	0.142515	0.137938	0.138936	0.042374	1.000000

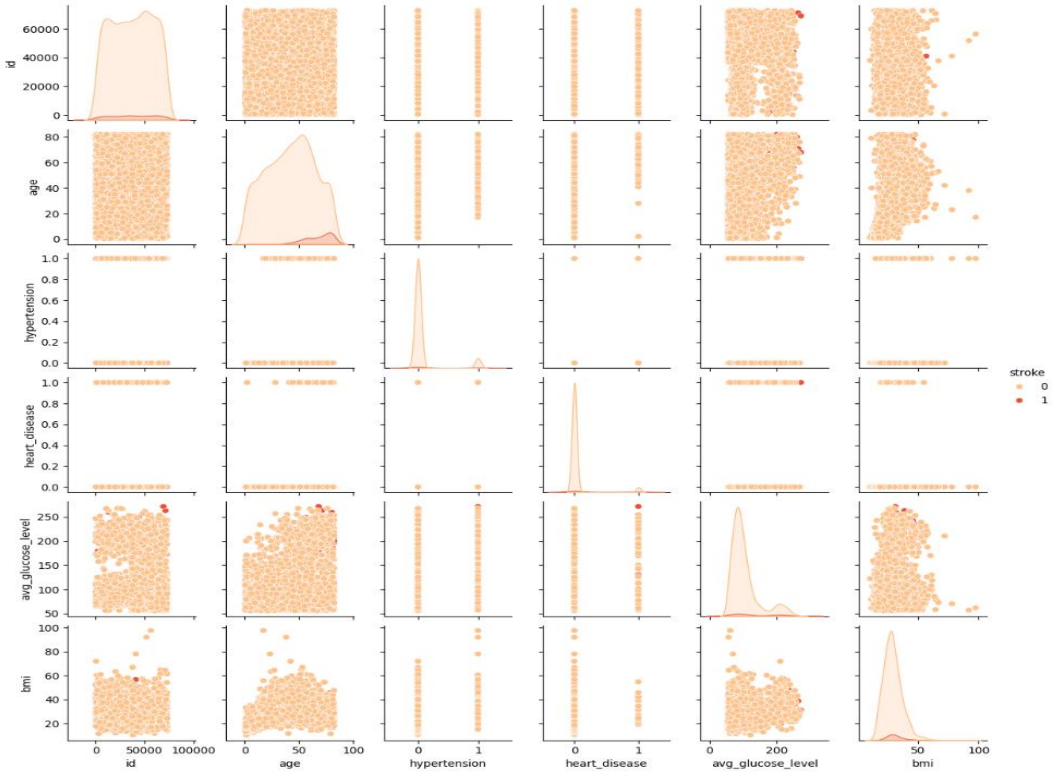


3. Boxplot

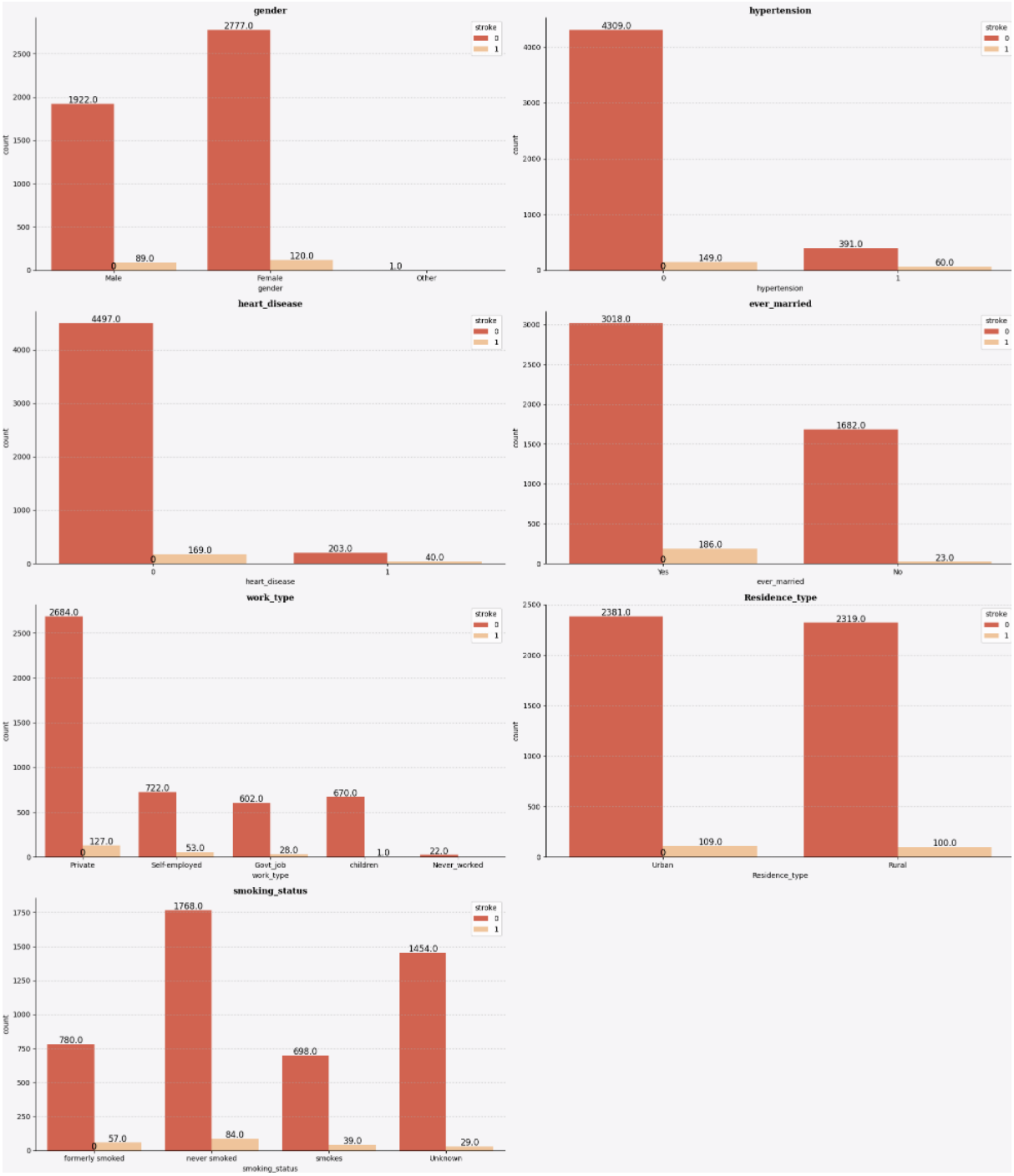
- Boxplots of the Numerical Variables
- The results suggest that the age of stroke patients tends to be older.
- BMI doesn't show any difference between stroke and non-stroke individuals, but outliers are present.
- Stroke patients generally have higher glucose levels, indicating a possible connection of diabetes with stroke.



4. **Pairplot**
   - The pair plot shows relationships numerical variables such as age, hypertension, heart disease, glucose level, and BMI with stroke cases.
   - Most stroke cases are associated with individuals who have hypertension or heart disease.
   - Higher glucose levels and BMI seem to only have a minor impact on stroke.



5. **Grouped Bar Chart**
   - Stroke cases across genders are relatively balanced.
   - Hypertension is a major risk for stroke as many individuals with hypertension have suffered with a stroke.
   - Individuals with heart disease have a significantly higher stroke risk.
   - Smoking and work type may have some influence.
   - Gender and residence type do not have a strong impact on stroke.


gender
stroke
0
1
count
2777.0
1922.0
89.0
120.0
1.0
Male
Female
Other
gender
hypertension
4309.0
149.0
391.0
60.0
0
1
hypertension
heart_disease
4497.0
169.0
203.0
40.0
heart_disease
ever_married
3018.0
186.0
1682.0
23.0
Yes
No
ever_married
work_type
2684.0
127.0
722.0
53.0
602.0
28.0
670.0
1.0
22.0
Private
Self-employed
Govt_job
children
Never_worked
work_type
Residence_type
2381.0
109.0
2319.0
100.0
Urban
Rural
Residence_type
smoking_status
780.0
57.0
1768.0
84.0
698.0
39.0
1454.0
29.0
formerly smoked
never smoked
smokes
Unknown
smoking_status

Data Mining Tasks

To prepare the data for machine learning, we focused on cleaning, transforming, and balancing the dataset so that our models could make more accurate and fair predictions.
We began by handling missing values. The BMI column had a few gaps, which we filled using the average BMI across all records. For the smoking status feature, many entries were labeled as "Unknown." We treated these as missing and replaced them with the most common smoking category in the dataset. We also removed the ID column since it was simply a unique identifier and didn't provide any useful information for prediction.
Next, we converted categorical variables into numerical formats. For features with only two values, like marital status and type of residence, we used label encoding. For features with more than two categories, such as gender, work type, and smoking status, we applied one hot encoding. This allowed us to preserve the meaning of each category without introducing any artificial order.
A key challenge we faced was the imbalance in the dataset. Since only a small percentage of people had experienced a stroke, most models tended to favor the majority class. To address this, we applied oversampling. This technique increased the number of stroke cases in the training data by duplicating examples from the minority class, which helped the models learn to recognize patterns related to stroke risk more effectively.
Finally, we scaled all numerical features using standardization so that they had similar ranges. This step was especially important for algorithms that are sensitive to feature magnitude.
Together, these data mining tasks helped us create a solid and reliable foundation for training our prediction models.

Data Mining Models/Methods

1. **Logistic Regression:**
   Logistic Regression is a widely used classification algorithm that predicts a binary outcome based on a set of input features. It works by applying a linear equation to the predictors and converting the output into probability. The predicted probability is then compared to a threshold to classify the outcome into one of two categories, in this case, stroke or no stroke.
   This model is particularly effective when the relationship between the independent variables and the target variable is linear.

   Advantages:
   i. Easy to interpret
   ii. Fast to train and efficient for large datasets
   iii. Well-suitable for linearly separable data

   Disadvantages:
   i. Cannot handle complex, non-linear relationships efficiently
   ii. Sensitive to outliers
   iii. Requires the independent variables to have a linear relationship

Implementation:
The Logistic Regression model put up a decent performance with an accuracy of **74.63%** and a ROC-AUC score of **74.80%**, showing that it's reasonably capable of distinguishing between stroke and no-stroke cases. It performed well in predicting the majority class (No Stroke) with an F1-score of **84.73%**, but struggled with the minority class (Stroke), achieving a low F1-score of **25.17%**. This indicates that the model tends to overlook stroke cases, a common issue with class imbalance. Since Logistic Regression is a linear model, it may not be capturing the complex relationships in the data.

2. **Decision Tree Classifier:**
   The Decision Tree Classifier is a popular supervised learning algorithm used for classification tasks. It works by recursively splitting the dataset into smaller subsets based on the feature that provides the highest information gain or reduces impurity the. At each decision node, the algorithm evaluates all possible splits and selects the one that results in the most homogeneous subsets. The final structure resembles a tree, where each internal node represents a decision based on a feature, and each leaf node represents the predicted class label.

   Advantages:
   i. Easy to understand and visualize
   ii. Can handle categorical and numerical data easily
   iii. Handles missing values effectively

   Disadvantages:
   i. Prone to overfitting
   ii. Sensitive to small changes in data
   iii. Struggles with complex data
   iv. Biased towards features with more unique values.

   Implementation:
   The Decision Tree Classifier, which works by splitting the data based on the most informative features, delivered a strong accuracy of **91.41%**. However, the ROC-AUC score of **53.84%**, the model is overfitting to the majority class. It achieved an impressive F1-score of **95.48%** for the no-stroke class but collapsed when it came to predicting strokes, with an F1-score of just **13.17%**. This is a clear sign that the model is memorizing the training data rather than generalizing patterns.

3. **Random Forest Classifier:**
   Random Forest is a powerful ensemble learning algorithm that improves predictive accuracy by combining the outputs of multiple decision trees. Unlike a single decision tree, which builds a single structure based on the entire dataset, Random Forest creates multiple decision trees using randomly selected subsets of both the data and features.

Each tree in the forest makes an independent prediction, and the final output is determined by majority voting for classification tasks. This ensemble approach helps reduce overfitting, improves stability, and enhances the model's ability to generalize to new data.

Advantages:
i. High predictive accuracy
ii. Does not overfit the model due to the structure of the model
iii. Effective for large dataset with mixed numerical and categorical features
iv. Can handle missing data without performance loss

Disadvantages:
i. Cannot handle complex since the model consist of multiple trees
ii. The model takes more time and cost
iii. Struggles with high dimensional data

Implementation:
The Random Forest Classifier, an ensemble model built from multiple decision trees, posted the highest accuracy at **94.01%**. But while it seems good at first, the ROC-AUC score of **52.29%** tells us that the model is not actually that good at distinguishing between classes. The F1-score for the majority class was outstanding at **96.91%**, but the minority class F1-score was bad at **9.01%**. This means it barely identified any strokes. The model is clearly overfitting to the majority class. Increasing the number of trees, reducing tree depth, and balancing the class weights could help address this imbalance and make the model more reliable

4. **XGBoost Classifier:**
XGBoost (Extreme Gradient Boosting) is a highly efficient and powerful ensemble learning algorithm that enhances the gradient boosting framework. It builds a series of decision trees sequentially, where each tree is designed to correct the errors made by the previous trees.
What sets XGBoost apart is its use of regularization techniques, which help prevent overfitting, and its ability to handle missing values by learning the best path for splits. The algorithm is also optimized for speed and performance, using parallel processing to accelerate training and reduce computational cost. These features make XGBoost a top choice for structured data and complex classification tasks.

Advantages:
i. High predictive accuracy
ii. Reduces the risk of overfitting
iii. Fast training process
iv. Can handle missing data without performance loss

Disadvantages:
i. Cannot handle complex model structures

ii. High cost
iii. Sensitive to hyperparameters

Implementation:
XGBoost showed a strong overall performance, with an accuracy of **92.47%** and the highest ROC-AUC score among the tree-based models at **56.37%**. It handled the majority class well with an F1-score of **96.05%**, but the minority class score of **19.10%** reveals that it still struggles with recall for stroke cases. The model's built-in regularization is helping prevent overfitting, but there can be improvement. Fine-tuning the learning rate, increasing tree depth, and using class balancing techniques could make XGBoost even more effective at detecting strokes.

5. **K-NN Classifier:**
The K-Nearest Neighbours (KNN) classifier is a simple and intuitive machine learning algorithm used for classification and regression tasks. It works by measuring the distance between a new data point and existing labelled data points, then assigning the label based on the majority class among the "k" closest neighbours
KNN relies on the assumption that similar data points exist close to each other in the feature space. The value of "k" determines the size of the voting group, which directly influences the model's complexity and sensitivity to noise. While KNN is easy to implement and understand, it can become computationally expensive for large datasets.

Advantages:
i. Easy to interpret and understand
ii. Works well for both regression and classification tasks
iii. Adapts well to small datasets

Disadvantages:
i. High memory usage
ii. Expensive model
iii. Sensitive to irrelevant and noisy features
iv. Struggles with high-dimensional data

Implementation:
KNN, which classifies data points based on their nearest neighbours, delivered an accuracy of **86.01%** and a solid ROC-AUC score of **59.30%**, the highest among the simpler models. It did a great job predicting the majority class with an F1-score of **92.34%**, but once again struggled with stroke cases, scoring only **19.18%** for the minority class. This suggests that KNN is influenced by the imbalance in the dataset, which is common since it relies heavily on the distance between data points. Scaling the data, adjusting the value of **k**, and experimenting with different distance metrics could help boost its sensitivity to the minority class.

6. **Support Vector Classifier:**
   Support Vector Classifier (SVC) is a supervised learning algorithm that aims to find the optimal boundary (or hyperplane) that separates data points into different classes. It works by identifying the hyperplane that maximizes the margin — the distance between the closest data points (support vectors) and the boundary. SVC is particularly effective in handling high-dimensional data and complex decision boundaries. When the data is not linearly separable, the algorithm uses kernel functions (such as linear, polynomial, and radial basis functions) to map the data into a higher-dimensional space where it becomes easier to create a separating hyperplane.

   Advantages:
   i. Effective for high dimensional data
   ii. Can handle both linear and non-linear data
   iii. Less prone to overfitting
   iv. Works well with small to medium sized datasets

   Disadvantages:
   i. Challenging to interpret higher dimensions
   ii. Expensive model which increases training time for larger dataset
   iii. Less effective on imbalanced data

   Implementation:
   The SVC, which works by finding the optimal hyperplane to separate data points, achieved an accuracy of **73.39%** and a ROC-AUC score of **75.12%**, one of the better scores for class separation. It performed well for the majority class with an F1-score of **83.83%**, but once again fell short on the minority class with an F1-score of **24.79%**. This suggests that the SVC is struggling with class imbalance, despite its strength in handling high-dimensional data. Trying different kernel functions, adjusting the regularization parameter, and using balanced class weights could help improve recall for the minority class.

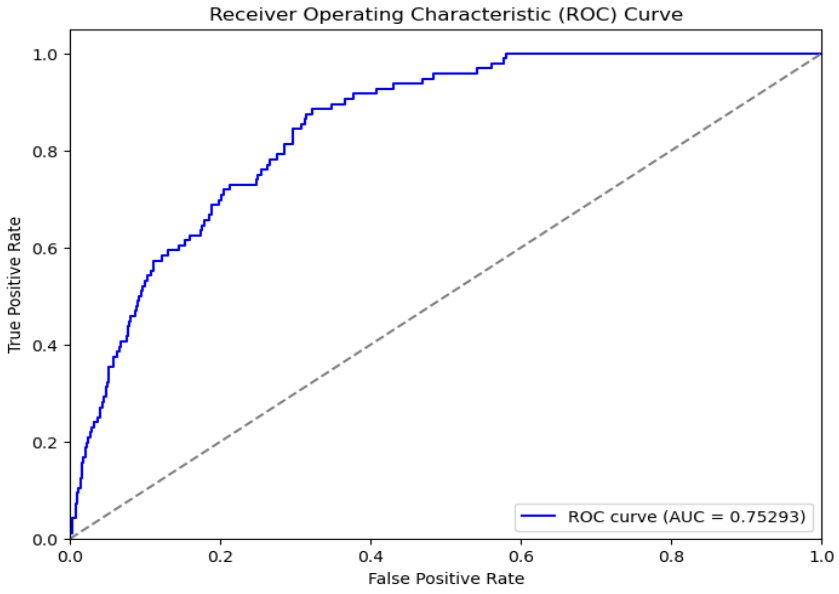
Performance Evaluation

1. **Logistic Regression**

   Logistic Regression is a widely used classification algorithm that predicts a binary outcome based on a set of input features. It works by applying a linear equation to the predictors and converting the output into a probability. The predicted probability is then compared to a threshold to classify the outcome into one of two categories, in this case, stroke or no stroke.

   This model is particularly effective when the relationship between the independent variables and the target variable is linear.

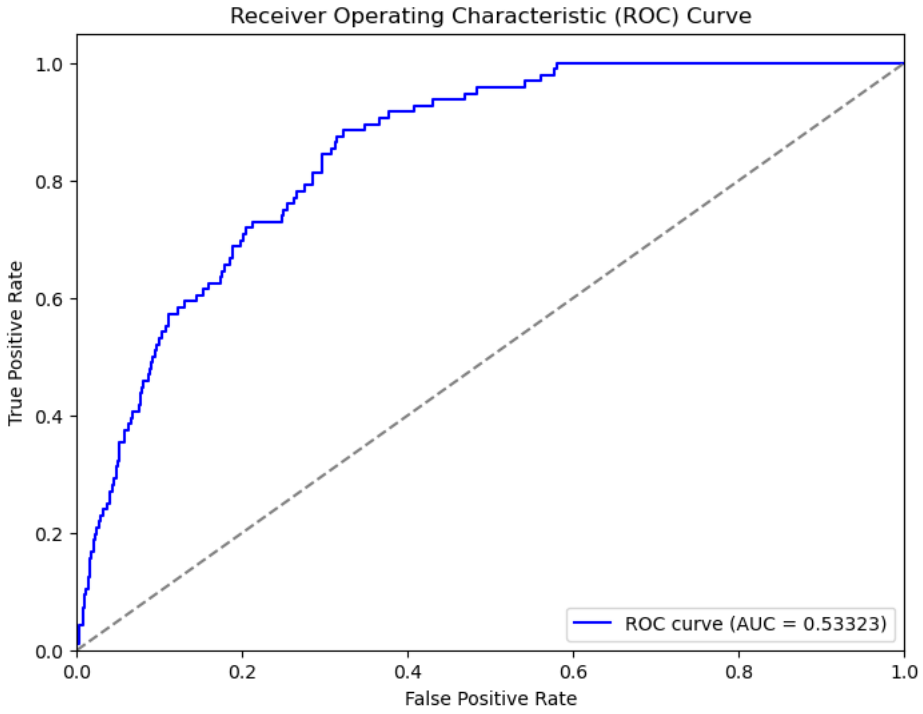
```
Logistic Regression

Confusion matrix:
 [[1186  405]
 [  23   73]]

Accuracy Score: 0.74630

ROC AUC Score: 0.75293

F1: 0.84714 0.25436
```



	Metric	Value
0	Accuracy	0.746295
1	Validation Error	0.253705
2	Sensitivity	0.760417
3	Specificity	0.745443
4	F1 Score	0.254355
5	ROC AUC	0.752930

2. **Decision Tree Classifier:**
   The Decision Tree Classifier is a popular supervised learning algorithm used for classification tasks. It works by recursively splitting the dataset into smaller subsets based on the feature that provides the highest information gain or reduces impurity the. At each decision node, the algorithm evaluates all possible splits and selects the one that results in the most homogeneous subsets. The final structure resembles a tree, where each internal node represents a decision based on a feature, and each leaf node represents the predicted class label.

```
Decision Tree Model

Confusion Matrix:
 [[1531   60]
 [  86   10]]

Accuracy Score: 0.91346

ROC AUC Score: 0.53323

F1: 0.95449 0.12048
```

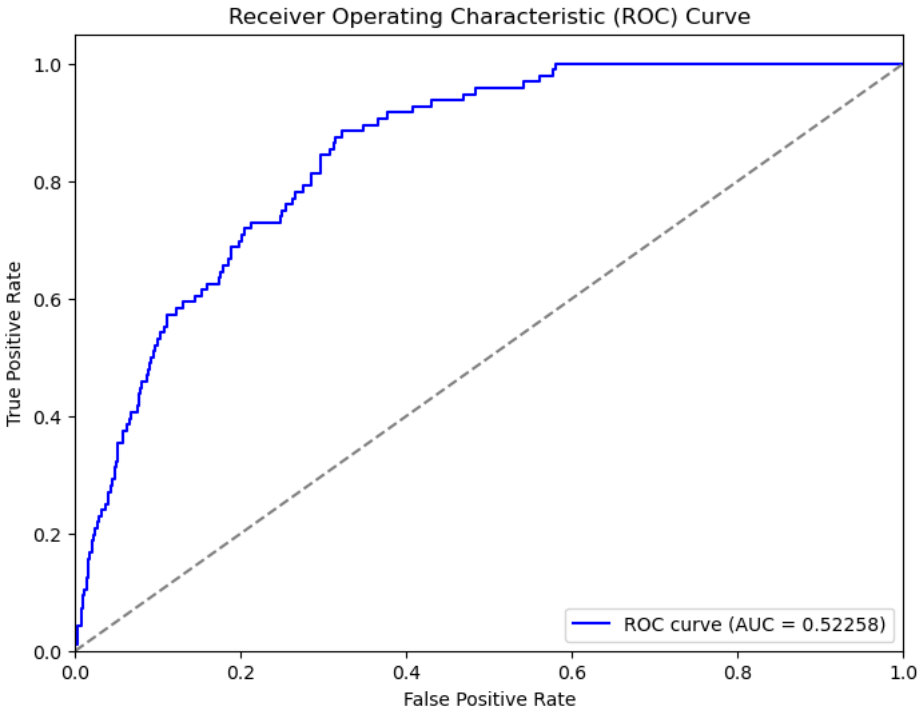


	Metric	Value
0	Accuracy	0.913456
1	Validation Error	0.086544
2	Sensitivity	0.104167
3	Specificity	0.962288
4	F1 Score	0.120482
5	ROC AUC	0.533227

3. **Random Forest Classifier:**
   Random Forest is a powerful ensemble learning algorithm that improves predictive accuracy by combining the outputs of multiple decision trees. Unlike a single decision tree, which builds a single structure based on the entire dataset, Random Forest creates multiple decision trees using randomly selected subsets of both the data and features.

   Each tree in the forest makes an independent prediction, and the final output is determined by majority voting for classification tasks. This ensemble approach helps reduce overfitting, improves stability, and enhances the model's ability to generalize to new data.

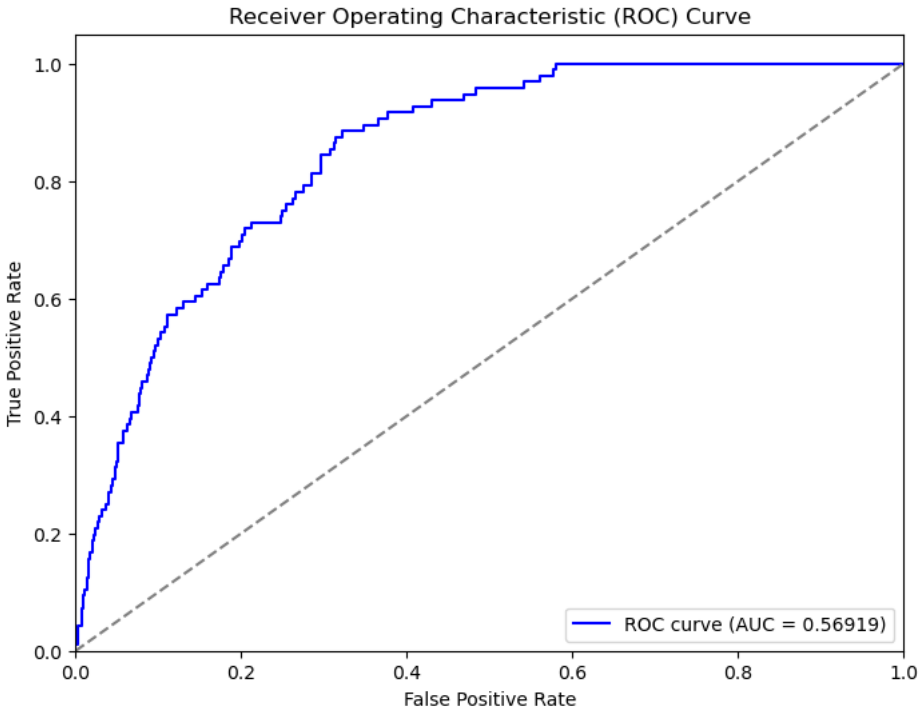
```
Random Forest

Confusion matrix:
 [[1580   11]
 [  91    5]]

Accuracy Score: 0.93954

ROC AUC Score: 0.52258

F1: 0.96873 0.08929
```



	Metric	Value
0	Accuracy	0.939538
1	Validation Error	0.060462
2	Sensitivity	0.052083
3	Specificity	0.993086
4	F1 Score	0.089286
5	ROC AUC	0.522585

4. **XGBoost Classifier:**
   XGBoost (Extreme Gradient Boosting) is a highly efficient and powerful ensemble learning algorithm that enhances the gradient boosting framework. It builds a series of decision trees sequentially, where each tree is designed to correct the errors made by the previous trees.
   What sets XGBoost apart is its use of regularization techniques, which help prevent overfitting, and its ability to handle missing values by learning the best path for splits. The algorithm is also optimized for speed and performance, using parallel processing to accelerate training and reduce computational cost. These features make XGBoost a top choice for structured data and complex classification tasks.

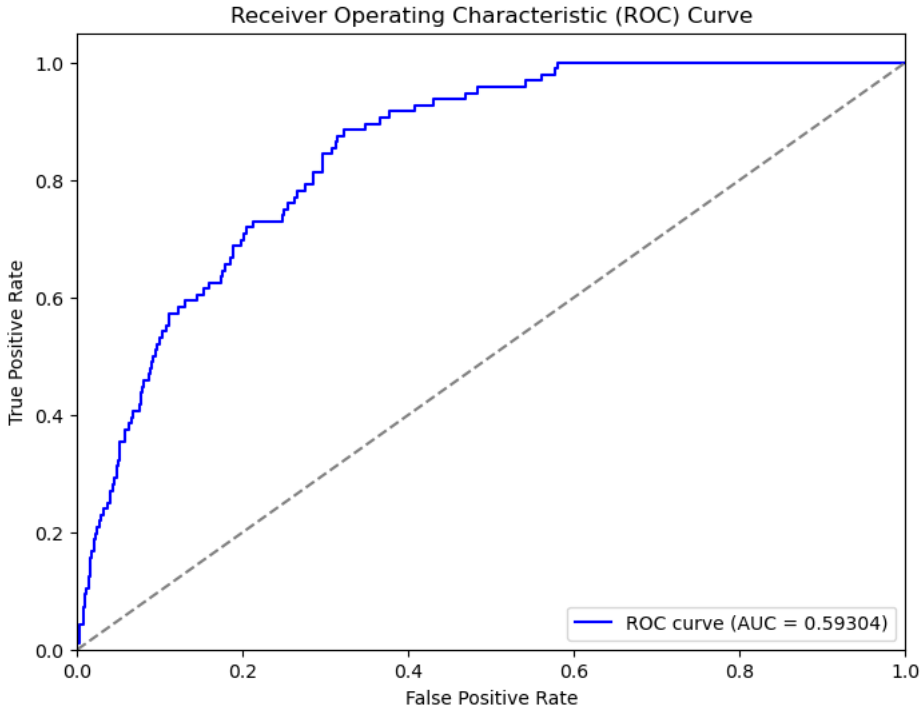
```
XGB

Confusion matrix:
 [[1546   45]
 [  80   16]]

Accuracy Score: 0.92590

ROC AUC Score: 0.56919

F1: 0.96114 0.20382
```



	Metric	Value
0	Accuracy	0.925904
1	Validation Error	0.074096
2	Sensitivity	0.166667
3	Specificity	0.971716
4	F1 Score	0.203822
5	ROC AUC	0.569191

5. **K-NN Classifier:**
   The K-Nearest Neighbours (KNN) classifier is a simple and intuitive machine learning algorithm used for classification and regression tasks. It works by measuring the distance between a new data point and existing labelled data points, then assigning the label based on the majority class among the "k" closest neighbours
   KNN relies on the assumption that similar data points exist close to each other in the feature space. The value of "k" determines the size of the voting group, which directly influences the model's complexity and sensitivity to noise. While KNN is easy to implement and understand, it can become computationally expensive for large datasets.

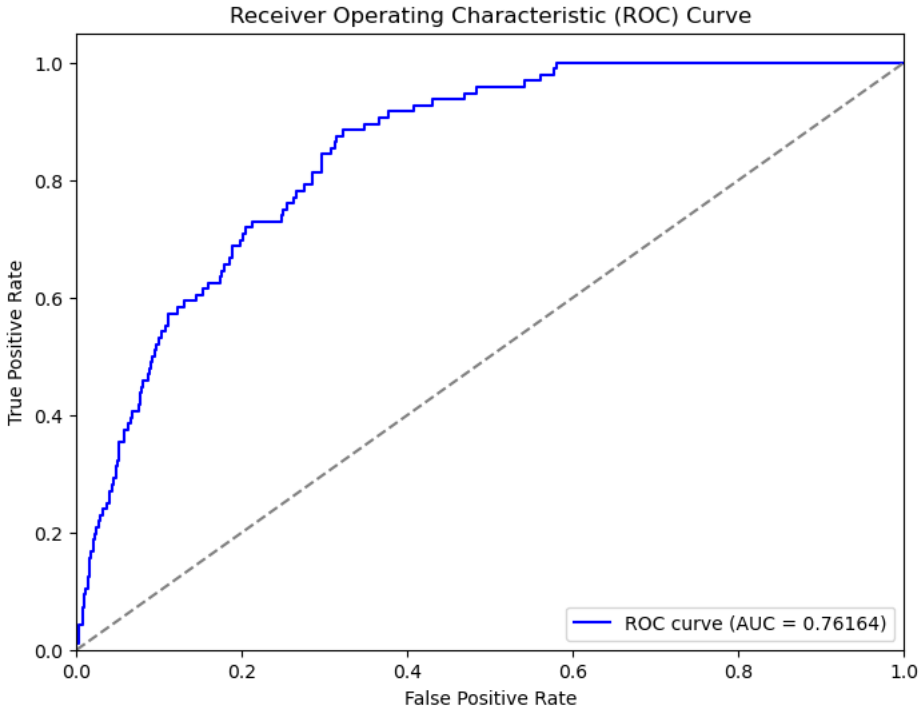
```
KNN Classifier

Confusion Matrix:
 [[1423  168]
 [  68   28]]

Accuracy Score: 0.86011

ROC AUC Score: 0.59304

F1: 0.92343 0.19178
```



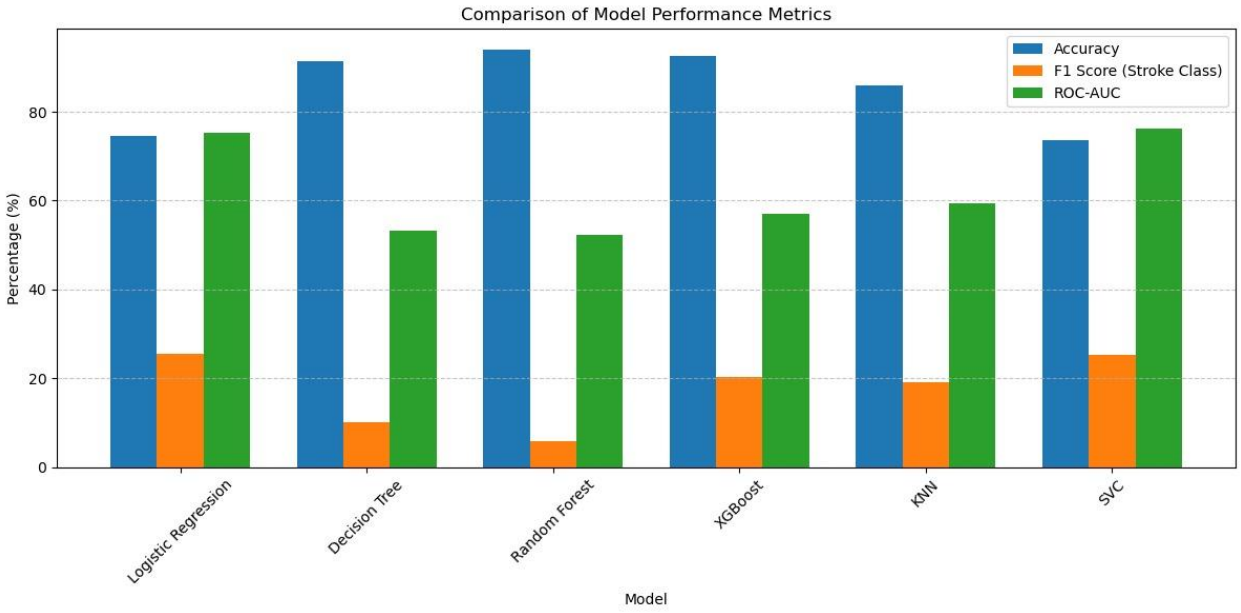
	Metric	Value
0	Accuracy	0.860107
1	Validation Error	0.139893
2	Sensitivity	0.291667
3	Specificity	0.894406
4	F1 Score	0.191781
5	ROC AUC	0.593036

6. **Support Vector Classifier:**
   Support Vector Classifier (SVC) is a supervised learning algorithm that aims to find the optimal boundary that separates data points into different classes. It works by identifying the hyperplane that maximizes the margin, the distance between the closest data points (support vectors) and the boundary. SVC is particularly effective in handling high-dimensional data and complex decision boundaries. When the data is not linearly separable, the algorithm uses kernel functions (such as linear, polynomial, and radial basis functions) to map the data into a higher-dimensional space where it becomes easier to create a separating hyperplane.

```
SVC Classifier

Confusion Matrix:
 [[1164  427]
 [  20   76]]

Accuracy Score: 0.73503

ROC AUC Score: 0.76164

F1: 0.83892 0.25376
```



	Metric	Value
0	Accuracy	0.735033
1	Validation Error	0.264967
2	Sensitivity	0.791667
3	Specificity	0.731615
4	F1 Score	0.253756
5	ROC AUC	0.761641



To better understand how the different models performed across key metrics, The above visual presents a side-by-side comparison of Accuracy, F1 Score for stroke cases, and ROC-AUC. While tree-based models such as Random Forest and Decision Tree achieved high accuracy, they significantly underperformed in identifying stroke cases, as reflected in their low F1 scores and ROC-AUC values. The Support Vector Classifier (SVC) offered the most balanced performance, achieving the highest ROC-AUC and a relatively strong F1 Score for stroke prediction. This highlights the importance of evaluating classification models using multiple metrics, particularly in imbalanced datasets where overall accuracy can be misleading.

Project Results

Logistic Regression: The logistic regression model performed reasonably well with an accuracy of 74.63 percent and a ROC AUC of 75.29 percent, showing it can moderately distinguish between classes. It had a balanced sensitivity of 76.04 percent and specificity of 74.54 percent, meaning it didn't favor one class too heavily. However, the F1 score was relatively low at 25.44 percent, highlighting that it struggles to maintain a strong balance between catching true positives and avoiding false positives. While it offers decent baseline performance, it misses a fair number of actual positive cases.

Decision Tree: The decision tree stood out for its high accuracy of 91.35 percent and excellent specificity of 96.23 percent, meaning it's great at identifying negatives. But its sensitivity was quite low at 10.42 percent, so it often failed to catch true positives. Its ROC AUC score of 53.32 percent and F1 score of just 12.05 percent show that despite its overall accuracy, it's not a reliable model for identifying minority class events, likely due to overfitting on the majority class.

Random Forest: Random Forest achieved the highest accuracy of all models at 93.95 percent and nearly perfect specificity at 99.31 percent, making it very precise when predicting negatives. However, like the decision tree, it performed poorly in detecting positive cases with a sensitivity of only 5.21 percent and an F1 score of 8.93 percent. Its ROC AUC of 52.26 percent reflects this

imbalance. In short, it's highly biased toward the majority class and is strong in precision but weak in recall.

XGBoost: XGBoost offered strong overall performance, with 92.59 percent accuracy, 97.17 percent specificity, and slightly better sensitivity than other tree-based models at 16.67 percent. Its ROC AUC of 56.92 percent and F1 score of 20.38 percent show modest improvements, but it still leans toward predicting negatives more confidently than positives. It performs better than Decision Tree and Random Forest at detecting positives but still has room to grow.

K-Nearest Neighbor: KNN showed a more balanced approach, with an accuracy of 86.01 percent, sensitivity of 29.17 percent, and specificity of 89.44 percent. Its ROC AUC of 59.30 percent and F1 score of 19.18 percent indicate it's better than the tree-based models at picking up positive cases, though still not ideal. It's a decent middle ground model that provides relatively stable results across all metrics.

SVC: SVC had the highest sensitivity at 79.17 percent, meaning it was the most effective at detecting positive cases, which is critical in imbalanced classification tasks. It also delivered the highest ROC AUC of 76.16 percent, showing strong ability to separate classes. While its accuracy was 73.50 percent and specificity was 73.16 percent, the F1 score of 25.38 percent reflects a solid trade-off between precision and recall. Overall, SVC is the most balanced and practical model when identifying true positives is the key objective.

	Model	Accuracy	Validation Error	Sensitivity	Specificity	F1 Score	ROC AUC
0	Logistic Regression	0.7463	0.2537	0.7604	0.7454	0.2544	0.7529
1	Decision Tree	0.9135	0.0865	0.1042	0.9623	0.1205	0.5332
2	Random Forest	0.9395	0.0605	0.0521	0.9931	0.0893	0.5226
3	XGBoost	0.9259	0.0741	0.1667	0.9717	0.2038	0.5692
4	KNN	0.8601	0.1399	0.2917	0.8944	0.1918	0.5930
5	SVC	0.7350	0.2650	0.7917	0.7316	0.2538	0.7616

Project Outcomes

In this project, we explored how data mining can help predict the risk of stroke using medical and demographic information from patients. We tested a range of classification models including Logistic Regression, Decision Tree, Random Forest, XGBoost, K Nearest Neighbors, and Support Vector Classifier to see which one could best identify people at risk.
Although models like Random Forest and XGBoost gave us very high accuracy, they struggled when it came to spotting actual stroke cases. This was mainly due to the imbalance in the dataset, where only a small number of patients had a stroke. These models were very good at predicting people without strokes but often missed those who were at risk.
The Support Vector Classifier stood out as the most balanced and effective model. It had the highest sensitivity at 79.17 percent, which means it was the best at correctly identifying stroke cases. It also had the strongest ROC AUC score of 76.16 percent and performed better than the other models in terms of overall reliability.

These findings show that in healthcare, it's not just about having a high accuracy score. What really matters is how well a model can catch the cases that need attention the most. In our case, the Support Vector Classifier proved to be the best at doing just that. It shows how data-driven tools like this can play a powerful role in supporting early intervention and improving patient outcomes.