Brain Stroke Analysis and Prediction

Akhil Pratyush Simhambhatla[1], Gokul Ragunandhan Narayanasamy[2], Venkata Mani Babu Karri [3]

Introduction:

The main objective of this project is to develop multiple machine-learning models to analyze the factors that are causing brain strokes in modern society. A brain stroke, as we know it, occurs suddenly, and may sometimes even cause death. It comes so unprecedented that it is difficult to identify the exact root cause for the issue sometimes. This brain condition occurs in all age groups irrespective of their physical and mental health. We wanted to build a system that analyzes some of the most prominent factors that might be a reason for the stroke to occur. We scouted the internet to get as much data as possible which is closer to our idea. We wanted to analyze physical features like Height, weight, BMI, gender, age, etc. along with other attributes like Marital status, type of work, and place of living. We successfully found one dataset which has all the necessary details mentioned above and started working on the data. The insights we got after analyzing the data were eye-opening and they revealed a lot of crucial information regarding stroke occurrences. As we go further into the in-depth analysis of the data, we figured out that a lot of factors are intertwined with each other, and their combination is sometimes even more dangerous. More information is provided as we dive deep into the analysis. We have taken the Machine Learning life cycle approach to handling this project.

Problem description:

Brain stroke is one of the primary reasons for death amongst different demographics across the globe. A stroke occurs suddenly and sometimes is caused by underlying reasons. Detecting a stroke and preventing it from happening is one of the most difficult tasks to do. Researchers, Neurologists, and other health advocates across the world have met together and made a list of guidelines that are the main reasons for brain stroke. The dataset we chose has multiple attributes that can help us in analyzing the sample data and coming up with a broad-spectrum conclusion that would help us in identifying the root cause of the illness. The project helped us to build a model that can let us see the underlying causes of stroke and it helps us peek at the environmental and psychological reasons that are causing stroke too. As a part of the project, we have trained and tested multiple machine learning models and used them to predict the occurrence of stroke based on the previous data. We used physical attributes as the focus features here as they have more impact on the human body than the rest. These models helped us identify the effects of age, gender, average blood glucose level, and BMI on the human body and their part in causing stroke.

Dataset Description:

The dataset we are using is the brain stroke dataset. This dataset has nearly 5000 unique rows belonging to different individuals. This data is intercepted from different hospitals to keep the data as balanced as possible and without any bias. It has basic information about the patient like age, gender, etc., along with that the dataset also gives us information about the crucial vitals of the patient like Body Mass Index (BMI), presence of Hypertension, marital status, work type, residence type, Average glucose levels, heart disease status and finally stroke status. This dataset would help us in relating various parameters and vital signs of a patient and analyzing this information to predict the occurrence of Brain Stroke in similar patients. The dataset can be found by clicking the link here. For your reference, 0 means the event has not occurred and 1 means the event has occurred. The dataset consists of both numerical and categorical data, but most of the data is divided into categories. So, we decided to take the categorical approach to analyze the data and to train and test the respective models available for categorical data.

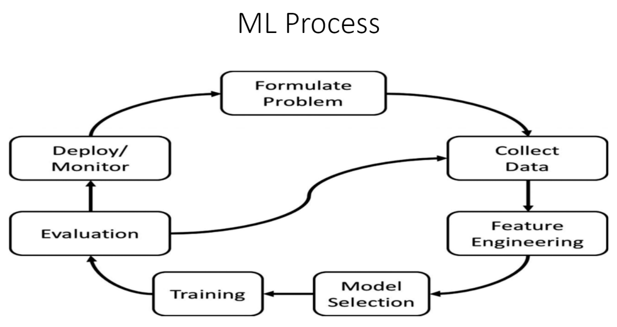


Fig 1. Machine learning life cycle

So far, we have covered Problem formulation and data collection. Let's see more about Data cleaning and Feature engineering now.

Data Cleaning:

Since we are dealing with categorical data here, we need to be sure that our data is completely clean, without any errors that might throw the results off. So, we paid keen attention to see if there are any potential data errors in the dataset we have. First things first, we converted the data from a CSV file to a data frame using pandas so that we can use python's inbuilt functionalities to clean the data. These are the goals of the data-cleaning process.

- Removing Zero values
- Removing negative values.
- Removing white spaces.
- Removing unnecessary punctuations.
- Removing unnecessary columns from the data.
- Encoding and indexing the data.
- Converting descriptive data into numerical data.
- Clustering the data.
- Dropping any unnecessary attributes.
- Removing unnecessary categories.

Negative Values and zero values will hamper the statistical analysis of the data a lot as these values will be greatly impacted by them. So that was given the top priority. Once they are removed, we removed any unnecessary columns like patient id, etc., which are not useful in our analysis. We then removed any unnecessary punctuations and white spaces from the data as they might cause issues while encoding and converting the data. We then removed any unnecessary classifications from the data which we thought were irrelevant to the analysis. For example, we thought of removing the smoking classification- unknown as it might not yield any valid information, so we removed that and some other attributes which didn't yield any valid information. After cleaning the data, we were left with 3500 entries with very a much smaller number of entries related to stroke. We decided to create a pair plot to see how the independent variables are related to the dependent variables in our dataset which can be seen in Fig 2.

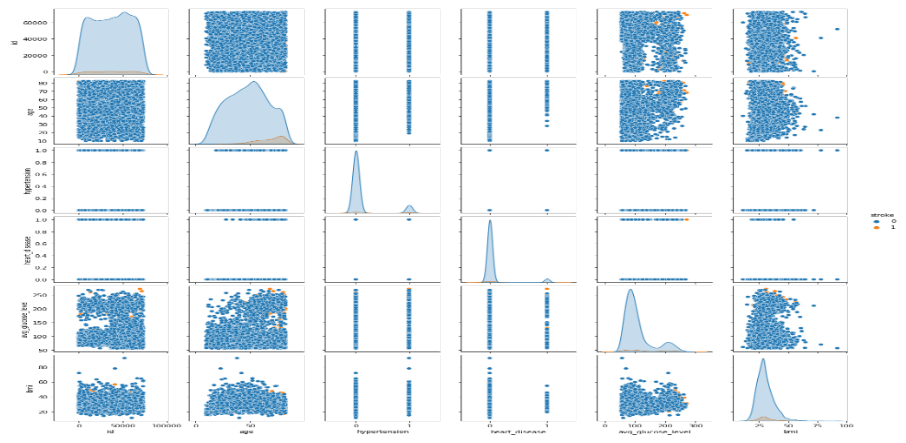


Fig 2: Pair plot between stroke and other physical parameters

As we see, the relationship between the dependent variable (stroke in orange dots) and the independent variables (all other variables in blue dots) is very and we were afraid that the dataset is completely imbalanced. So, we decided to up sample the data and add the stroke victims to a believable number so that the machine learning models can be trained using balanced data rather than imbalanced data. We up sampled the data to an extent that after this process, we got a 50-50 ratio of normal and stroke patients. This up sampling is done randomly so that the data would stay true. A pair plot is drawn post up sampling which can be seen in Fig 3.

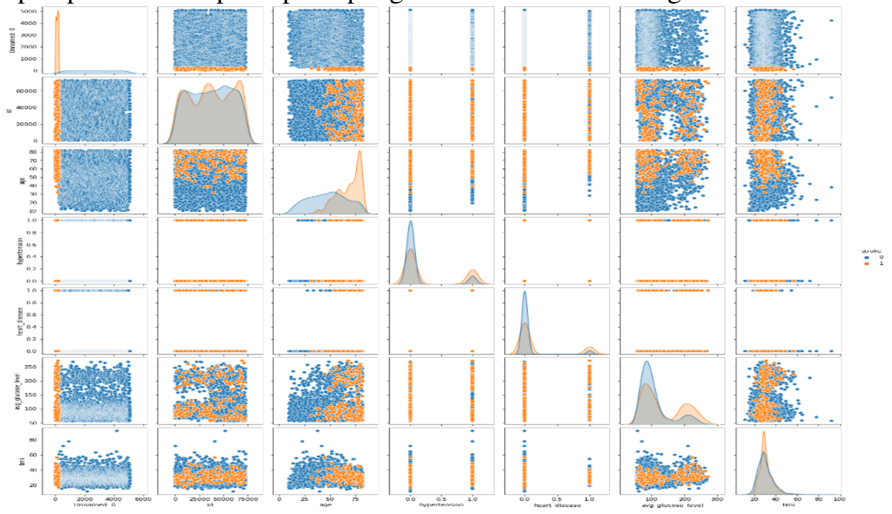


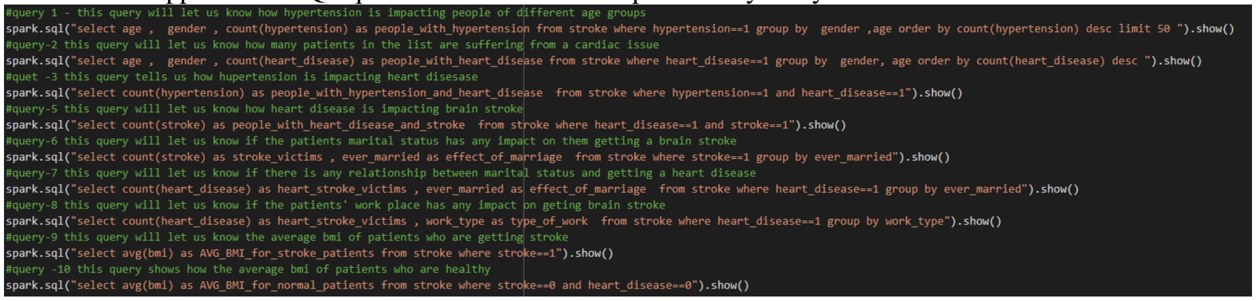
Fig 3: Pair Plot post Up sampling

As we can see how the data looks more balanced and the relationship between the dependent and the independent variables has improved multi fold. We ended up with a total of 6727 rows.

Methodologies: We decided to take a Three Prong approach to analyze the data we had. The more analysis we did, the more findings we got from the data. These are the methods we used to analyze the data. Each method used is discussed in detail further.

- Data Analysis using Spark SQL.
- Exploratory Data Analysis using python functions.
- Fitting multiple models for the dataset using python inbuilt functions and Spark MLlib.

1. Spark SQL Analysis: We know that SQL has a lot of potential in terms of summarizing and analyzing huge datasets. We thought of using Spark SQL initially to get some preliminary questions answered and a rudimentary analysis is done before getting into the actuality of the project. Through Spark SQL, we successfully found out the relationship between each parameter in the dataset and how it is impacting the stroke in real-time. We came up with 17 SQL queries that helped us in developing a basic idea about what we will be dealing with later. Fig 4. Shows us a snippet of the SQL queries we used to do the preliminary analysis of the data.



```
#query 1 - this query will let us know how hypertension is impacting people of different age groups
spark.sql("select age ,  gender , count(hypertension) as people_with_hypertension from stroke where hypertension==1 group by  gender ,age order by count(hypertension) desc limit 50 ").show()
#query-2 this query will let us know how many patients in the list are suffering from a cardiac issue
spark.sql("select age ,  gender , count(heart_disease) as people_with_heart_disease from stroke where heart_disease==1 group by  gender, age order by count(heart_disease) desc ").show()
#quet -3 this query tells us how hupertension is impacting heart disesase
spark.sql("select count(hypertension) as people_with_hypertension_and_heart_disease  from stroke where hypertension==1 and heart_disease==1").show()
#query-5 this query will let us know how heart disease is impacting brain stroke
spark.sql("select count(stroke) as people_with_heart_disease_and_stroke  from stroke where heart_disease==1 and stroke==1").show()
#query-6 this query will let us know if the patients marital status has any impact on them getting a brain stroke
spark.sql("select count(stroke) as stroke_victims , ever_married as effect_of_marriage  from stroke where stroke==1 group by ever_married").show()
#query-7 this query will let us know if there is any relationship between marital status and getting a heart disease
spark.sql("select count(heart_disease) as heart_stroke_victims , ever_married as effect_of_marriage  from stroke where heart_disease==1 group by ever_married").show()
#query-8 this query will let us know if the patients' work place has any impact on geting brain stroke
spark.sql("select count(heart_disease) as heart_stroke_victims , work_type as type_of_work  from stroke where heart_disease==1 group by work_type").show()
#query-9 this query will let us know the average bmi of patients who are getting stroke
spark.sql("select avg(bmi) as AVG_BMI_for_stroke_patients from stroke where stroke==1").show()
#query -10 this query shows how the average bmi of patients who are healthy
spark.sql("select avg(bmi) as AVG_BMI_for_normal_patients from stroke where stroke==0 and heart_disease==0").show()
```

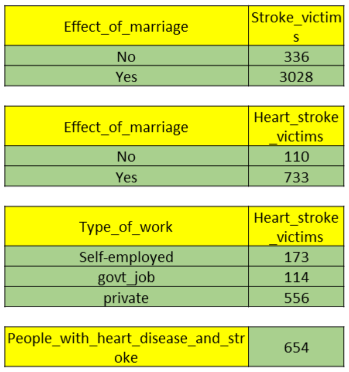
Fig 4: SQL Queries for Spark SQL Analysis

These are some of the findings we got from this rudimentary analysis of the data.

- Female Population is suffering more from stroke than the male population.
- People having heart stroke are more susceptible to getting brain stroke.
- People getting brain stroke have more BMI than normal people.

- People with more Hypertension are also falling into the stroke category.
- People with high blood sugar are also victims of brain stroke.
- People in their ages ranging from 50-80 are more affected by stroke.
- Smoking also has an important role in causing stroke.
- People who are married are more susceptible to stroke.
- People who are working in Private firms tend to get strokes more than people working for government firms.

Fig 5 shows us some example outputs of the SQL analysis. More outputs can be seen in the code. Results of Spark SQL analysis are shown in Fig-18.



Effect_of_marriage	Stroke_victims
No	336
Yes	3028

Effect_of_marriage	Heart_stroke_victims
No	110
Yes	733

Type_of_work	Heart_stroke_victims
Self-employed	173
govt_job	114
private	556

People_with_heart_disease_and_stroke	654

Fig 5: Sample outputs

2. Exploratory Data Analysis using python functions: We know that python is used to mine small to large amounts of data and get the required information, we already did the basic data exploration with SQL, and here we do the advanced exploratory analysis such as looking for correlations through each other parameters through multiple, data visualizations such as pie charts, bar graphs, and correlations matrix. We used matplotlib to visualize to do the data analysis in python. It is a graph plotting library in python that is used for visualization. It is an open-source library and mostly built on python.

These are some of the findings we got from this analysis of the data.

- The dataset has a Population with hypertension of around 20%.
- The dataset has a Population with heart disease of around 13%.
- More than 80% of the population in the dataset is married people.
- Around 60% of the population in the dataset is employed in the private sector.
- Around 50% of the population in the dataset never had a smoking habit.
- Around 50% of the population in the dataset had been identified with stroke.

Since smoking is identified to be one of the important factors causing stroke, we visualized the identification of stroke between various people with varying smoking habits. Fig 7. Shows us a snippet of the stroke identification distribution within the smoking status parameter. Below there are some of the findings we got from this analysis of the data.

- The data shows that the major population who formerly smoked had been identified as stroke patients.
- The data shows that half the population who are smoking currently had been identified already as stroke patients.
- The data shows that the major population who never smoked had been not identified as stroke patients.

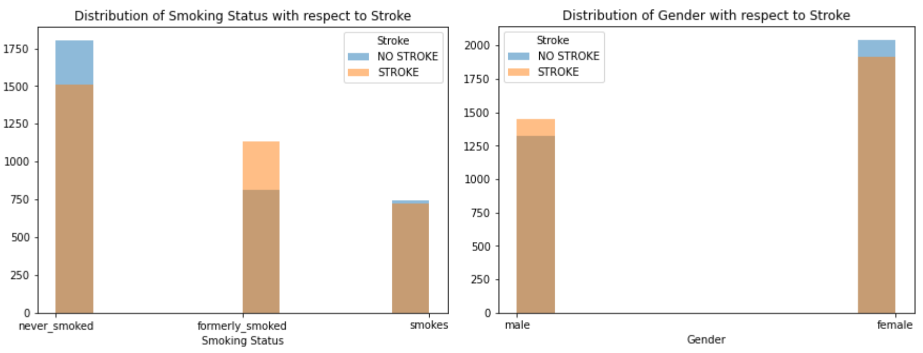


Fig 7 and 8: Smoking status, Gender vs Stroke Identification

We also tried to visualize the impact of the stroke with respect to gender. Fig 8. Shows us a snippet of the stroke identification with respect to gender. We could see that major of the male population in the dataset has been identified with stroke. At last, we tried to visualize the impact of the stroke with respect to work status. Fig 9. Shows us a snippet of the stroke identification with respect to work status. Below there are some of the findings we got from this analysis of the data.

- Almost 50% of the population in the dataset who are working in the private sector and government jobs has been identified with stroke compared to other sectors.
- More than 50% of the population in self-employment had been identified with stroke.
- People who are the age of children and who never worked have not been identified with any stroke cases.

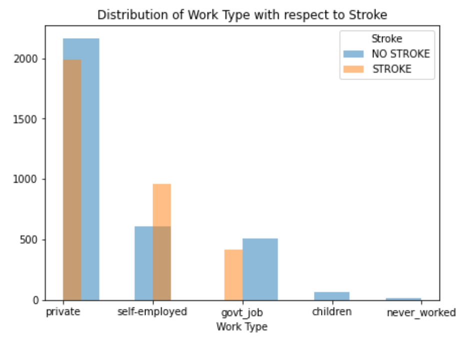


Fig 9: Work Type vs Stroke Identification

Correlation Matrix:

A correlation matrix is a table that displays the coefficients of correlation between parameters in a dataset. Each cell in the table represents the relationship between two parameters' values. A correlation matrix can be used to summarize data, which can be useful to do more advanced analysis. Fig 9. Shows us the correlation matrix of all the parameters in the dataset. Below there are some of the findings we got from the correlation matrix.

- Hypertension, Age of the patient, Blood glucose level, and Smoking habit have a higher correlation with stroke when compared to other parameters in the dataset.
- Population who never married, the population under the children category, and the population who never smoked had very weak chances of getting a stroke. Fig-10: Correlation Matrix

Building Machine Learning Models using python:

We used the sci-kit learn library from python to build machine learning models. Scikit-learn is an open-source Machine learning analysis library and a standard library for Machine Learning (ML). Important features of sci-kit learn are simple and efficient features for data mining and analysis. It features most of the classification, regression, and clustering algorithms including SVM, random forests, gradient boosting, k-means, etc. It's built on top of NumPy, SciPy, and matplotlib. It's free and commercially used everywhere. Here we picked up five of the best-performing and widely used models to see the performance accuracy of each model and pick the best model for our dataset. The models used are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, K nearest neighbors, and Naïve Bayes Classifier.

Logistic Regression: Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. Logistic regression is an analysis method for classification problems, where you are trying to determine if a new sample fits best into a category. Since all the data pre-processing is done previously, here we split the dataset into test and train data into 70% and 30% proportions. Where 70% of the dataset is used to train the model and 30% of the dataset is used to test the model's accuracy by predicting the known outcomes. The commonly used metrics to measure the accuracy of machine learning models are explained below:
Confusion Matrix: It is an N **x** N matrix where N is the number of classes being predicted. Since the output is of two categories, we get the N value as 2. Commonly used measuring matrices in a confusion matrix are listed below.

- Accuracy: the ratio of the total number of predictions that were predicted correctly.
- Positive Predictive Value or Precision: the proportion of positive outputs that were correctly identified.
- Negative Predictive Value: the proportion of negative outputs that were predicted correctly.
- Sensitivity or Recall: the proportion of actual positive outputs that were predicted correctly.
- Specificity: the proportion of actual negative outputs that were predicted correctly.

F1- Score: The F-score, also called the F1-score, It is mainly used to evaluate binary classification systems, which classify the output in either a positive or negative group. The formula for calculating F1-score is shown below:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

The Logistic Regression model has an accuracy score of 0.747 and the results are shown in Fig-19.
Decision Tree Classifier: Decision Trees are nonparametric supervised learning method which is built upon iteratively asking questions to partition data. The aim of the tree is to increase the predictiveness as much as possible at each partitioning so that model gains more information. It is suitable to work on a mix of feature data types continuous, categorical, and binary. The model has an overall accuracy of 0.97. The results are shown in Fig-19.
Random Forest Classifier: It is an ensemble of many decision trees. It is used in methods called bagging where decision trees are used as parallel estimators. Here the result is based on the majority vote of the results received from each decision tree. It uses the mean value of results from the decision trees. Also, it computes in parallel so that time does not become a bottleneck. This model is widely used for high dimensional data sets compared to fast linear models. The results listed below show that the model has an overall accuracy of 0.98. The results are shown in Fig-19.
Naive Bayes: It is a supervised algorithm used for classification hence it is also called a "Naïve Bayes Classifier". It assumes features are independent of each other and there is no correlation between features. The naïve assumption of features being uncorrelated is the reason why this algorithm is called "Naive". The intuition behind the naive is Bayer's theorem. It is a very fast processing algorithm but less accurate compared to other algorithms. The results of the model have an overall accuracy of 0.75. The results are shown in Fig-19.
Background about Spark MLlib: The Apache Spark MLlib is a machine learning library used to carry out machine learning operations. Spark MLlib is composed of four main parts: First, algorithms 2. Pipelines 3. Featurization, followed by 4. Utilities There are many different machine learning (ML) algorithms, including clustering, regression, and classification models. Additionally, the pipeline for constructing and running models in MLlib. Its four primary duties. Constructing, Evaluating, Featurization, and Pipelines are the first three methods that assist us in performing multiple activities in a single phase. Featurization is the process by which we take the raw data and turn it into useful features. It accommodates various data types. Labeled point, distributed matrix, local matrix, and local vector. Row Matrix, Indexed Row Matrix, Coordinate Matrix, and Block Matrix are the data formats that Spark MLlib supports for distributed matrices. The RDD-based API (spark. mllib package) and the dataframe-based API are available in Spark 2.0. (spark.ml package). Because RDD-based one is in maintenance mode and MLlib is migrating to Dataframe-based API due to its benefits, we are using the spark.ml package for this project. Additionally, MLlib includes statistical and linear algebra tools that are beneficial when performing various statistical studies.
MLlib Pipeline: The main component of utilizing MLlib to execute models is called MLlib Pipeline. Let's look at what the MLlib pipeline can accomplish.

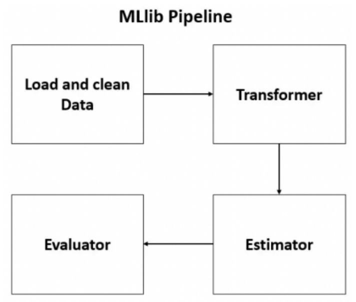
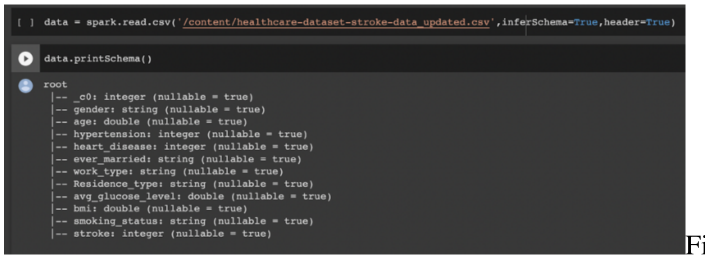


Fig-11: MLlib Pipeline Flow Chart

1. Spark Session: The first step is creating a spark session to start programming with dataset using spark.

2.Spark Data frame Operations: In this step, we read the dataset into a dataframe. Then perform some operations on the dataframe like defining schema, count etc.

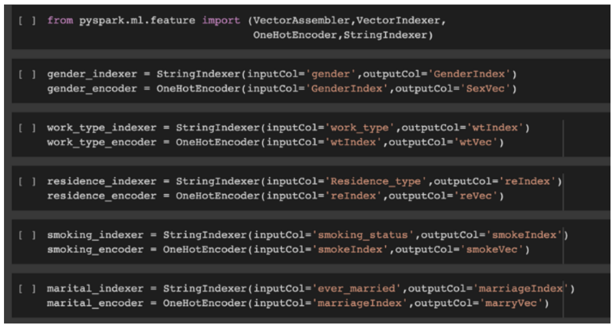
3. Data Exploration: Data Exploration is the step where we explore our dataset, and identify things like data dimensions, identifying missing values, data description, unique values etc. Then taking required action based on this information to make our data ready. Since we already did this and updated our dataset, we can skip this part here



```
[ ] data = spark.read.csv('/content/healthcare-dataset-stroke-data_updated.csv',inferSchema=True,header=True)

data.printSchema()

root
 |-- _c0: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- age: double (nullable = true)
 |-- hypertension: integer (nullable = true)
 |-- heart_disease: integer (nullable = true)
 |-- ever_married: string (nullable = true)
 |-- work_type: string (nullable = true)
 |-- Residence_type: string (nullable = true)
 |-- avg_glucose_level: double (nullable = true)
 |-- bmi: double (nullable = true)
 |-- smoking_status: string (nullable = true)
 |-- stroke: integer (nullable = true)
```

Fig-12: Defining Schema, Count, Columns

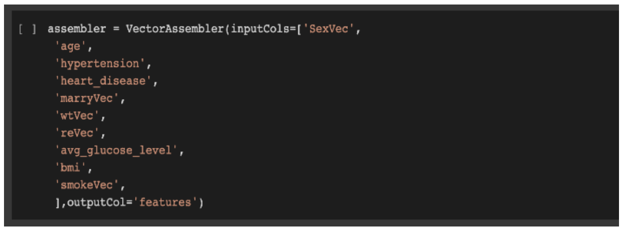
4. Encoding Categorical Variables: As we all know most of the ML algorithms works with numerical data. So it is important to convert all categorical values into numerical form and encode them. We cannot directly encode categorical values, first need to convert them into numeric. String Indexer takes a categorical column has Input Column and returns a numerical column as an output column, gender, work_type, Residence_type, smoking_status and ever_married as categorical columns. We apply string indexer on them first. We encode the above generated numeric columns using one hot encoder into sparse vectors.



```
[ ] from pyspark.ml.feature import (VectorAssembler,VectorIndexer,
                                    OneHotEncoder,StringIndexer)

[ ] gender_indexer = StringIndexer(inputCol='gender',outputCol='GenderIndex')
    gender_encoder = OneHotEncoder(inputCol='GenderIndex',outputCol='SexVec')

[ ] work_type_indexer = StringIndexer(inputCol='work_type',outputCol='wtIndex')
    work_type_encoder = OneHotEncoder(inputCol='wtIndex',outputCol='wtVec')

[ ] residence_indexer = StringIndexer(inputCol='Residence_type',outputCol='reIndex')
    residence_encoder = OneHotEncoder(inputCol='reIndex',outputCol='reVec')

[ ] smoking_indexer = StringIndexer(inputCol='smoking_status',outputCol='smokeIndex')
    smoking_encoder = OneHotEncoder(inputCol='smokeIndex',outputCol='smokeVec')

[ ] marital_indexer = StringIndexer(inputCol='ever_married',outputCol='marriageIndex')
    marital_encoder = OneHotEncoder(inputCol='marriageIndex',outputCol='marryVec')
```

Fig-13: String Indexing, One Hot Encoding

5. Vector Assembler: Vector Assembling assemble all updated columns and other columns into a single vector column. Here we are naming our output column as ‘features’.



```
[ ] assembler = VectorAssembler(inputCols=['SexVec',
     'age',
     'hypertension',
     'heart_disease',
     'marryVec',
     'wtVec',
     'reVec',
     'avg_glucose_level',
     'bmi',
     'smokeVec',
     ],outputCol='features')
```

Fig-14: Vector Assembling

6. Building ML Pipeline: After defining the above steps we choose a model for our data and input feature columns and label columns (stroke) and split our data into train and test data. Here we used 70% of data for training and 30% for testing. Then, we build ML Pipelines input each transformation as a stage into the pipeline, and our people. ML Pipeline applies all the transformations in a single step and fit the model on our data. Trains it and Tests it and finally evaluates it.

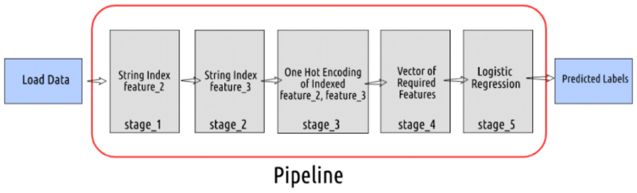
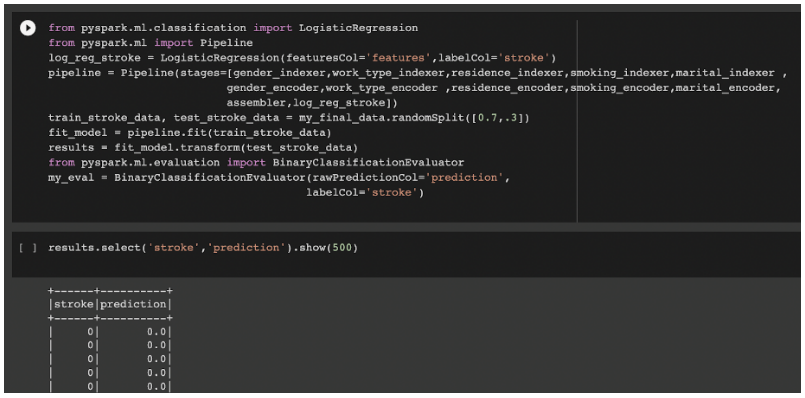


Fig-15: ML Pipeline Working



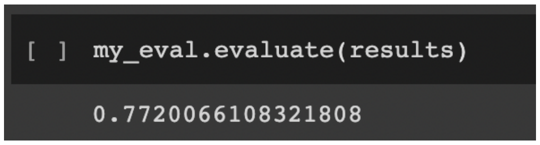
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
log_reg_stroke = LogisticRegression(featuresCol='features',labelCol='stroke')
pipeline = Pipeline(stages=[gender_indexer,work_type_indexer,residence_indexer,smoking_indexer,marital_indexer ,
                            gender_encoder,work_type_encoder ,residence_encoder,smoking_encoder,marital_encoder,
                            assembler,log_reg_stroke])
train_stroke_data, test_stroke_data = my_final_data.randomSplit([0.7,.3])
fit_model = pipeline.fit(train_stroke_data)
results = fit_model.transform(test_stroke_data)
from pyspark.ml.evaluation import BinaryClassificationEvaluator
my_eval = BinaryClassificationEvaluator(rawPredictionCol='prediction',
                                        labelCol='stroke')
```

```
[ ] results.select('stroke','prediction').show(500)
```

```
+------+----------+
|stroke|prediction|
+------+----------+
|     0|       0.0|
|     0|       0.0|
|     0|       0.0|
|     0|       0.0|
|     0|       0.0|
```

Fig-16: Logistic Regression using Spark MLlib

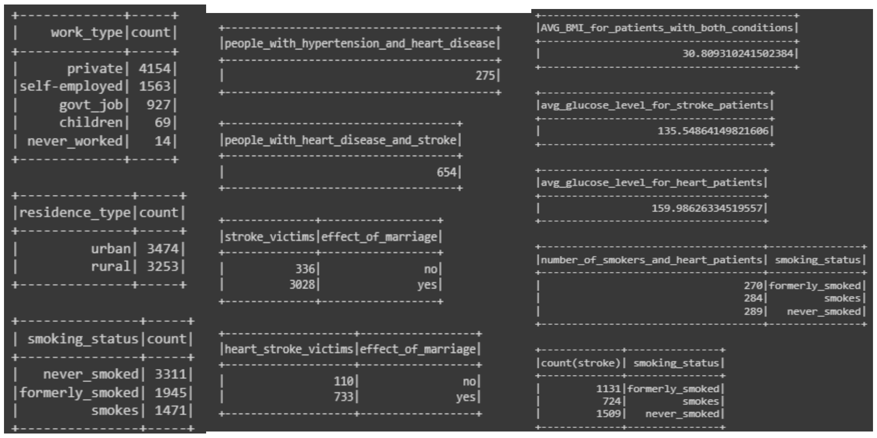
7. Evaluation: Evaluating the accuracy of model based on prediction values using BinaryClassificationEvaluator. Accuracy value for Logistic Regression model is shown below.



```
[ ] my_eval.evaluate(results)

    0.7720066108321808
```

Fig-17: Calculating Accuracy

Results: SQL Results:



```
+-------------+-----+
|    work_type|count|
+-------------+-----+
|      private| 4154|
|self-employed| 1563|
|     govt_job|  927|
|     children|   69|
| never_worked|   14|
+-------------+-----+

+--------------+-----+
|residence_type|count|
+--------------+-----+
|         urban| 3474|
|         rural| 3253|
+--------------+-----+

+---------------+-----+
| smoking_status|count|
+---------------+-----+
|   never_smoked| 3311|
|formerly_smoked| 1945|
|         smokes| 1471|
+---------------+-----+
```

```
+-------------------------------------+
|people_with_hypertension_and_heart_disease|
+-------------------------------------+
|                                  275|
+-------------------------------------+

+--------------------------------+
|people_with_heart_disease_and_stroke|
+--------------------------------+
|                             654|
+--------------------------------+

+--------------+-----------------+
|stroke_victims|effect_of_marriage|
+--------------+-----------------+
|           336|               no|
|          3028|              yes|
+--------------+-----------------+

+--------------------+-----------------+
|heart_stroke_victims|effect_of_marriage|
+--------------------+-----------------+
|                 110|               no|
|                 733|              yes|
+--------------------+-----------------+
```

```
+----------------------------------------+
|AVG_BMI_for_patients_with_both_conditions|
+----------------------------------------+
|                      30.809310241502384|
+----------------------------------------+

+-------------------------------------+
|avg_glucose_level_for_stroke_patients|
+-------------------------------------+
|                    135.54864149821606|
+-------------------------------------+

+------------------------------------+
|avg_glucose_level_for_heart_patients|
+------------------------------------+
|                    159.98626334519557|
+------------------------------------+

+-----------------------------------+---------------+
|number_of_smokers_and_heart_patients| smoking_status|
+-----------------------------------+---------------+
|                                270|formerly_smoked|
|                                284|         smokes|
|                                289|   never_smoked|
+-----------------------------------+---------------+

+-------------+---------------+
|count(stroke)| smoking_status|
+-------------+---------------+
|         1131|formerly_smoked|
|          724|         smokes|
|         1509|   never_smoked|
+-------------+---------------+
```

Fig 18: SQL Analysis results

Scikit-learn based models results:

Logistic regression				Decision Tree			
Category	**precision**	**recall**	**f1-score**	**Category**	**precision**	**recall**	**f1-score**
No Stroke	0.77	0.72	0.74	No Stroke	1	0.93	0.97
Stroke	0.73	0.77	0.75	Stroke	0.94	1	0.97
accuracy			0.75	accuracy			0.97
macro avg	0.75	0.75	0.75	macro avg	0.97	0.97	0.97
veighted av	0.75	0.75	0.75	weighted avg	0.97	0.97	0.97

Random Forest				Naïve Bayes			
Category	**precision**	**recall**	**f1-score**	**Category**	**precision**	**recall**	**f1-score**
No Stroke	1	0.97	0.98	No Stroke	0.8	0.6	0.73
Stroke	0.96	Random	0.98	Stroke	0.71	0.82	0.76
accuracy			0.98				
macro avg	0.98	0.98	0.98	accuracy			0.75
veighted av	0.98	0.98	0.98	macro avg	0.75	0.75	0.75
				weighted avg	0.75	0.75	0.75

Final Accuracy	
Model	**Accuracy**
Decision Tree	0.965
Random Forest	0.972
Logistic Regression	0.747
Naïve Bayes	0.748

Fig-19 Final Results

Based on all the above-explained models we could see that the top two accuracy score models either a decision tree or random forest can be used to identify strokes in our dataset since they have an accuracy of 0.97.

Spark MLlib Models Results: After training, testing, and evaluating all four models. These are the results we got.

Model	**Accuracy (%)**
Logistic Regression	76.99
Random Forest	77.13
Naïve Bayes	71.87
Decision Tree	78.59

Fig-20

Conclusion: Finally, we successfully updated our dataset. Built ML pipelines trained and tested various machine learning models from Spark MLlib to predict brain stroke based on the other features. We also observed differences in the accuracy score between models used from scikit-learn and models used from Spark MLlib library.

Appendix 1: Individual Contributions:

Akhil – Dataset Collection, Data Cleaning, and Spark SQL analysis, Report writing.

Gokul – Python analysis, Scikit learn Machine Learning models, and Report writing.

Venkata Mani- Spark MLlib Analysis, Report writing.

References:

- "Machine Learning Library (MLlib) Guide", https://spark.apache.org/docs/latest/ml-guide.html
- "Pythonic Data cleaning using Numpy and Pandas", https://realpython.com/python-data-cleaning-numpy-pandas/
- "Data Cleaning in Python", https://towardsdatascience.com/data-cleaning-in-python-the-ultimate-guide-2020-c63b88bf0a0d
- "Data Cleaning with Python", https://monkeylearn.com/blog/data-cleaning-python/
- "Data Analysis using Spark SQL", https://www.analyticsvidhya.com/blog/2021/08/an-introduction-to-data-analysis-using-spark-sql/
- "Spark SQL, DataFrames and Datasets Guide", https://spark.apache.org/docs/latest/sql-programming-guide.html
- "Understanding Spark SQL", https://www.edureka.co/blog/spark-sql-tutorial/
- "Upsampling and Downsampling Imbalanced Data in Python", https://wellsr.com/python/upsampling-and-downsampling-imbalanced-data-in-python/
- "Python | Pandas DataFrame", https://www.geeksforgeeks.org/python-pandas-dataframe/
- "Scikit-learn(sklearn) in Python", https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/
- "Learning Model Building in Scikit-learn", https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/
- "Machine Learning Project in Python Step-By-Step", https://machinelearningmastery.com/machine-learning-in-python-step-by-step/
- "scikit-learn", https://scikit-learn.org/stable/
- "Scikit-learn Tutorial", https://www.dataquest.io/blog/sci-kit-learn-tutorial/
- "How to use Confusion Matrix in Scikit-Learn (with Example)", https://www.jcchouinard.com/confusion-matrix-in-scikit-learn/
- "Evaluation Metrics - spark.mllib", https://home.apache.org/~pwendell/spark-nightly/spark-branch-1.6-docs/spark-2.0.0-SNAPSHOT-2016_06_05_11_01-4e767d0-docs/mllib-evaluation-metrics.html
- "Build Machine Learning pipelines" https://www.analyticsvidhya.com/blog/2019/11/build-machine-learning-pipelines-pyspark/
- "Spark ML Lib, ML Pipelines", https://medium.com/analytics-vidhya/spark-mllib-6f70a7062e3b
- "Apache Spark Machine Learning", https://data-flair.training/blogs/spark-mllib-data-types/
- "Machine Learning Library (MLlib) Guide", https://spark.apache.org/docs/2.2.0/ml-guide.html
- "OneHotEncoder and Pipelines", https://medium.com/@nutanbhogendrasharma/role-of-onehotencoder-and-pipelines-in-pyspark-ml-feature-part-2-3275767e74f0
- "VectorAssembler in PySpark", https://pyshark.com/vectorassembler-in-pyspark/