STROKE DEVELOPMENT PREDICTOR

Project 4 Presentation by Meme, Emily, Sherry and Hiam

February 19th, 2022

- 15 million people worldwide suffer a stroke.
- A stroke occurs when blood supply to parts of the brain is interrupted or reduced, which prevents brain tissue from getting it's needed nutrients.
- It is commonly seen amongst those with high blood pressure, diabetes, obesity, and smokers of age 40 years and above mostly.[2]

BREAKDOWN

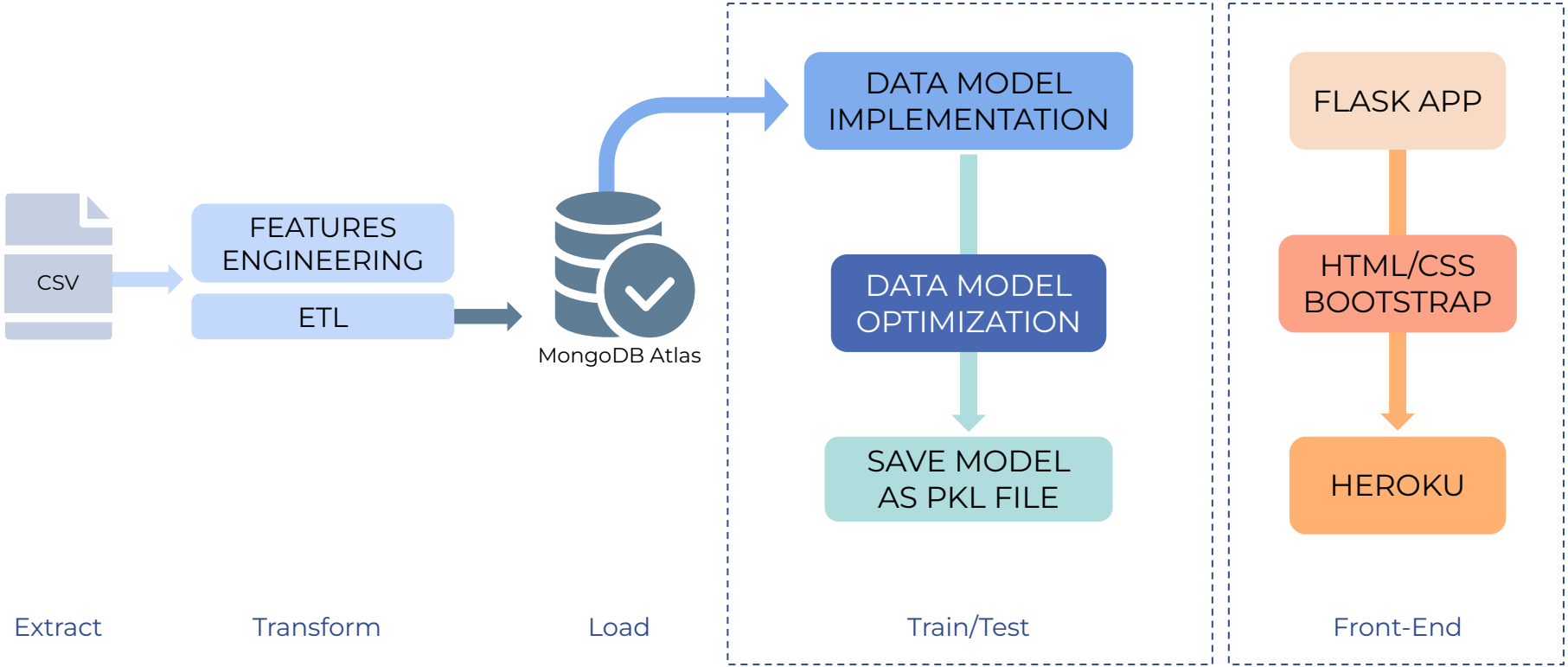
01

Project Scope

Choose the most Optimized Supervised Classification Machine Learning Model

Create an Interactive Application where User Input is Used for Model Testing/Prediction

ARCHITECTURE OF PROCESS



02

Exploratory Data Analysis

EXPLORING OUR FEATURES

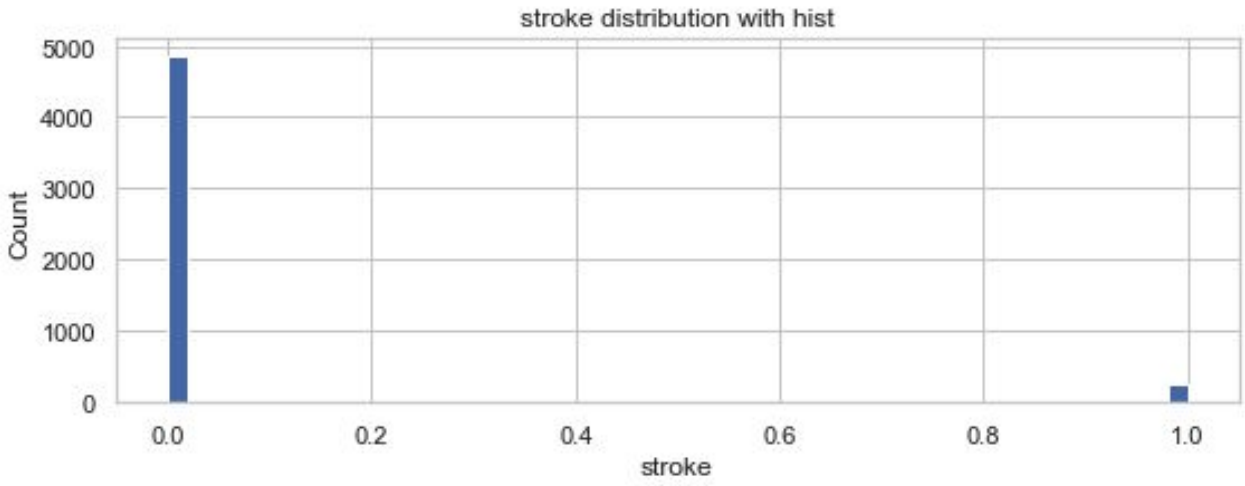
01 Understanding the features and outputs provided

02 Identifying and analyzing our target Feature vs labelled Features

03 Looking into features correlation

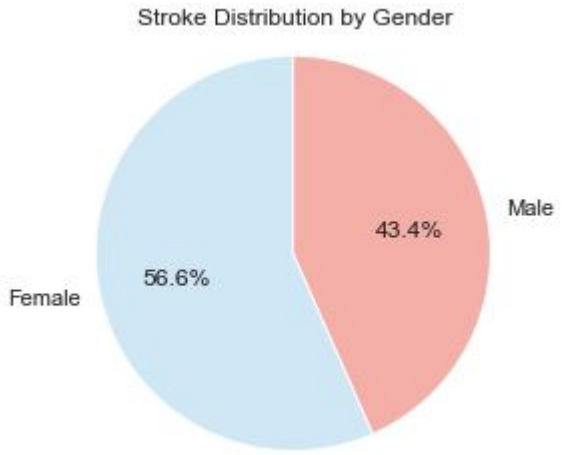
04 Cleaning the data and Loading on MongoDB Atlas

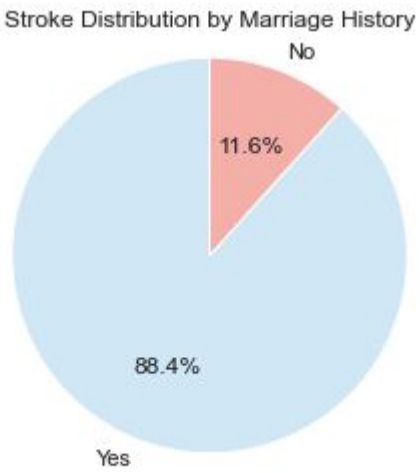
TARGET FEATURE COUNT: STROKE

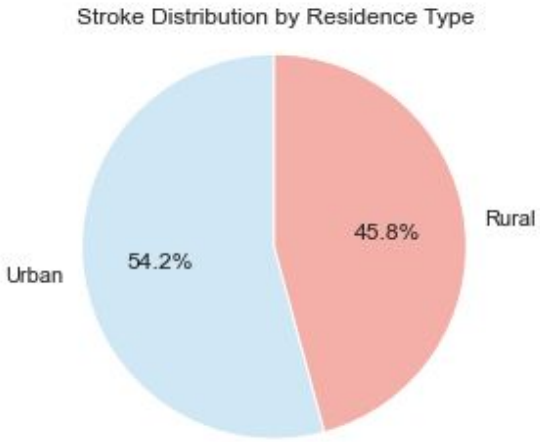



```
stroke:
0    4861
1     249
Name: stroke, dtype: int64
```

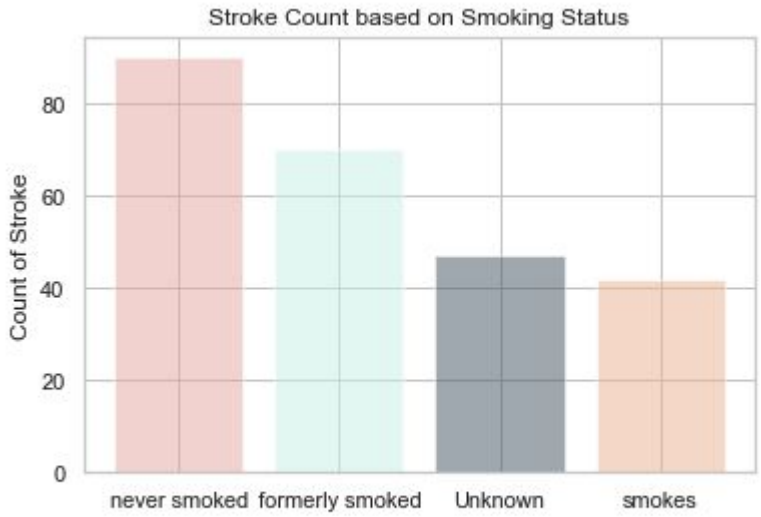
CATEGORICAL FEATURES VS STROKE (1)

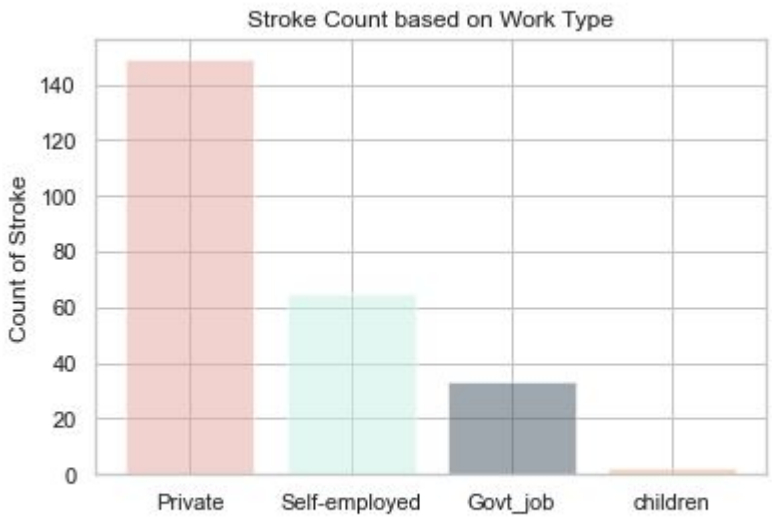




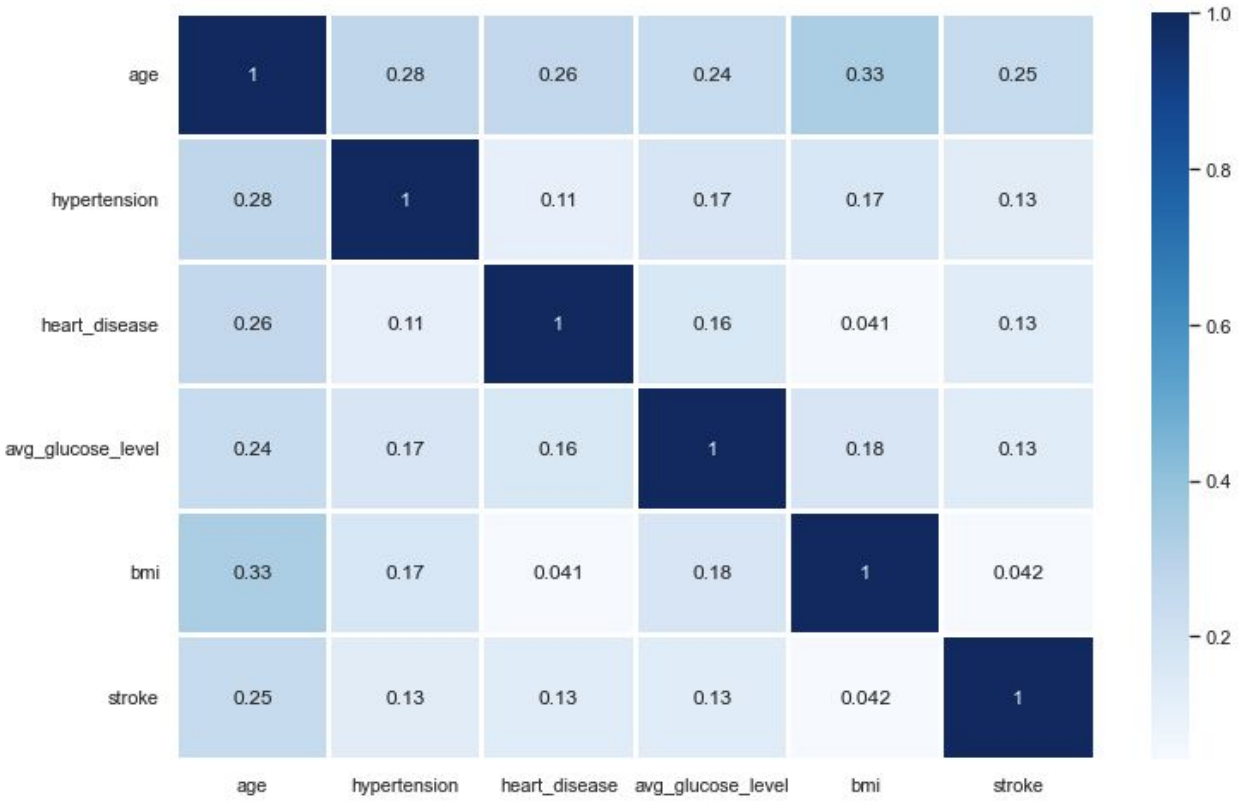


CATEGORICAL FEATURES VS STROKE (2)





CORRELATION



DATA CLEANING

Removing ID column

Anomaly Detection

Remove outliers from dataset

The BMI feature contained over 150 Null Values

Identifying Null Values

Replacing Null Values with Appropriate Values

Replace null values with average value BMI based on gender

The 'Other' gender only had 1 observation

Removed 'Other' Gender

MongoDB Atlas

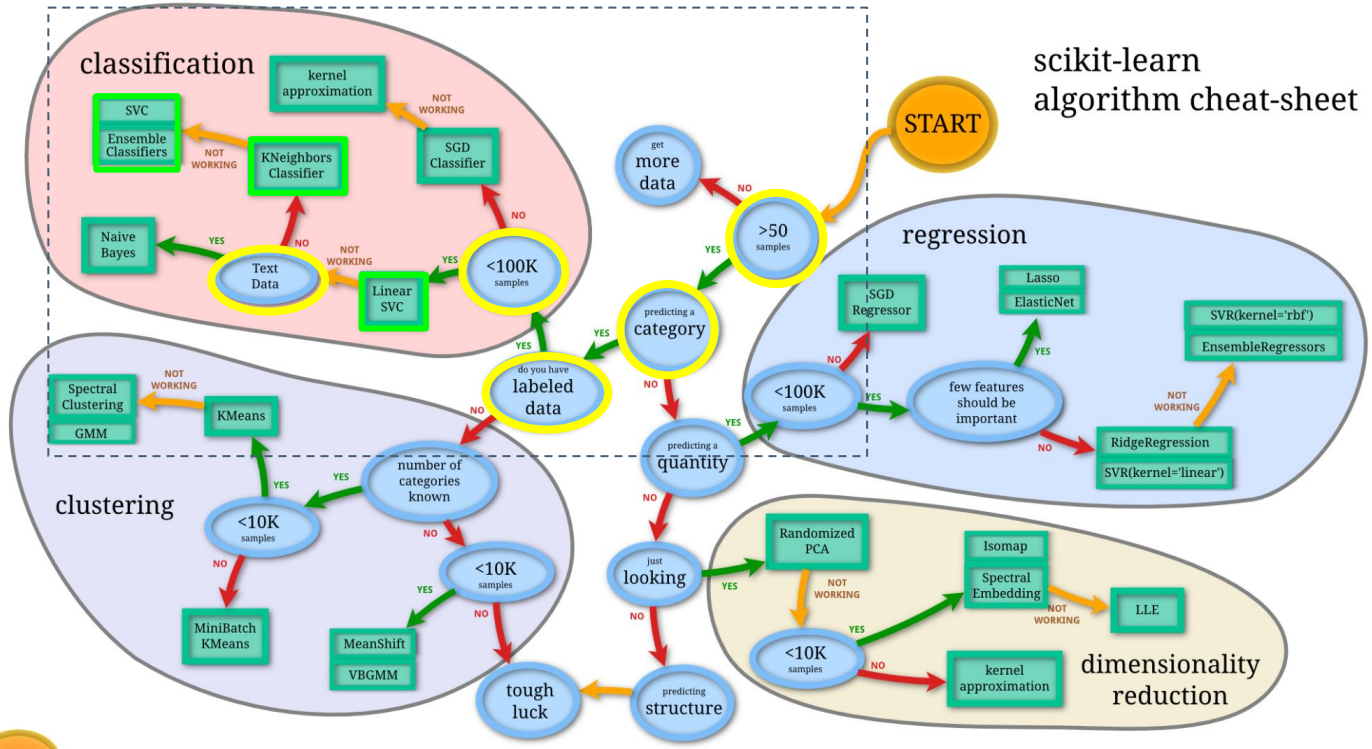
DATA LABELING

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	29.035926	never smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
3	Male	81.0	0	0	Yes	Private	Urban	186.21	29.000000	formerly smoked	1

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	0	61.0	0	0	1	3	0	202.21	29.035926	2	1
1	1	80.0	0	1	1	2	0	105.92	32.500000	2	1
2	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
3	1	81.0	0	0	1	2	1	186.21	29.000000	1	1

03

Data Model Implementation


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
NOT WORKING
Text Data
YES
<100K samples
Linear SVC
get more data
NO
>50 samples
YES
predicting a category
YES
do you have labeled data
YES
NO
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
Spectral Clustering
GMM
NOT WORKING
KMeans
NO
number of categories known
YES
YES
<10K samples
NO
<10K samples
YES
NO
MiniBatch KMeans
MeanShift
VBGMM
clustering
predicting a quantity
YES
NO
just looking
NO
YES
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
<10K samples
YES
NO
kernel approximation
dimensionality reduction
tough luck
predicting structure
Back
scikit learn

MODELS EXPLORED

Support Vector Machine (SVM)

K-Nearest Neighbor (KNN)

Random Forest Classifier (RF)

Decision Tree Classifier (DT)

FIRST MODEL IMPLEMENTATION

SVM

	precision	recall	f1-score	support
0	0.96	1.00	0.98	962
1	0.00	0.00	0.00	45
accuracy			0.96	1007
macro avg	0.48	0.50	0.49	1007
weighted avg	0.91	0.96	0.93	1007

KNN

	precision	recall	f1-score	support
0	0.96	1.00	0.98	962
1	0.20	0.02	0.04	45
accuracy			0.95	1007
macro avg	0.58	0.51	0.51	1007
weighted avg	0.92	0.95	0.93	1007

Random Forest

	precision	recall	f1-score	support
0	0.94	0.98	0.96	507
1	0.98	0.93	0.96	500
accuracy			0.96	1007
macro avg	0.96	0.96	0.96	1007
weighted avg	0.96	0.96	0.96	1007

Decision Tree

	precision	recall	f1-score	support
0	0.94	0.90	0.92	978
1	0.90	0.94	0.92	957
accuracy			0.92	1935
macro avg	0.92	0.92	0.92	1935
weighted avg	0.92	0.92	0.92	1935

FIRST MODEL IMPLEMENTATION

SVM

	precision	recall	f1-score	support
0	0.96	1.00	0.98	962
1	0.00	0.00	0.00	45
accuracy			0.96	1007
macro avg	0.48	0.50	0.49	1007
weighted avg	0.91	0.96	0.93	1007

KNN

	precision	recall	f1-score	support
0	0.96	1.00	0.98	962
1	0.20	0.02	0.04	45
accuracy			0.95	1007
macro avg	0.58	0.51	0.51	1007
weighted avg	0.92	0.95	0.93	1007

Random Forest

	precision	recall	f1-score	support
0	0.94	0.98	0.96	507
1	0.98	0.93	0.96	500
accuracy			0.96	1007
macro avg	0.96	0.96	0.96	1007
weighted avg	0.96	0.96	0.96	1007

Decision Tree

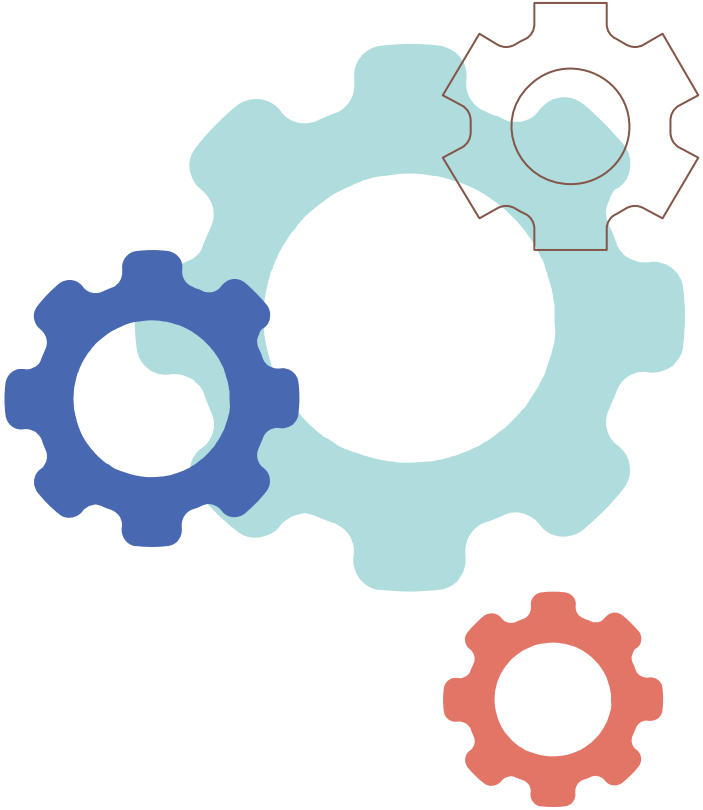
	precision	recall	f1-score	support
0	0.94	0.90	0.92	978
1	0.90	0.94	0.92	957
accuracy			0.92	1935
macro avg	0.92	0.92	0.92	1935
weighted avg	0.92	0.92	0.92	1935

Pre-Optimization

Accuracy: **95.70%**
Recall: **93.40%**
F1-Score: **95.59%**

04

Data Model Optimization



Optimization Methods (1)

SMOTE

```
from imblearn.over_sampling import SMOTE
#Define indpendent and dependent variables - and remove the variable to be predicted
X = en_dataset.drop('stroke', axis=1)
y = en_dataset['stroke']
smote = SMOTE()
X,y = smote.fit_resample(X,y)
```

```
0    4837
1     197
Name: stroke, dtype: int64
```

→

```
1    4837
0    4837
Name: stroke, dtype: int64
```

GridSearchCV Optimizer

```
# defining parameter range
param_grid = {'n_neighbors': [1,3,5,7,9,11,15,19],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
             'weights': ['uniform', 'distance'],
             'metric': ['manhattan', 'euclidean', 'minkowski', 'cosine', 'jaccard', 'hamming']}

grid = GridSearchCV(knn_model, param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train_scaled, y_train)
```

```
{'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 2, 'weights': 'uniform'}
KNeighborsClassifier(metric='manhattan', n_neighbors=2)
```

Optimization Methods (2)

Scaling Data

```
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.transform(X_test)
```

Stratify

Before

```
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
print(Counter(y_train))
print(Counter(y_test))
# Not balanced when the test size is 0.5 this is imbalanced data
# Counter({0: 4794, 1: 306})
# Counter({0: 2391, 1: 159})
# Counter({0: 2403, 1: 147})
```

After

```
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42, stratify=y)
print(Counter(y_train))
print(Counter(y_test))
# Balanced when test_size is 50%
# Counter({0: 4794, 1: 306})
# Counter({0: 2397, 1: 153})
# Counter({0: 2397, 1: 153})
```

Final Optimization RFC Model

Attempt #2 - Optimization Method: ***Stratify***

	precision	recall	f1-score	support
0	0.93	0.97	0.95	504
1	0.97	0.93	0.95	503
accuracy			0.95	1007
macro avg	0.95	0.95	0.95	1007
weighted avg	0.95	0.95	0.95	1007

Attempt #3 - Optimization Methods: **SMOTE** + **StandardScaler**

Attempt #4 - Optimization Methods: **SMOTE** + **StandardScaler + Stratify**

Attempt #5 - Optimization Methods: **SMOTE** + **GridSearchCV** + ***Stratify***

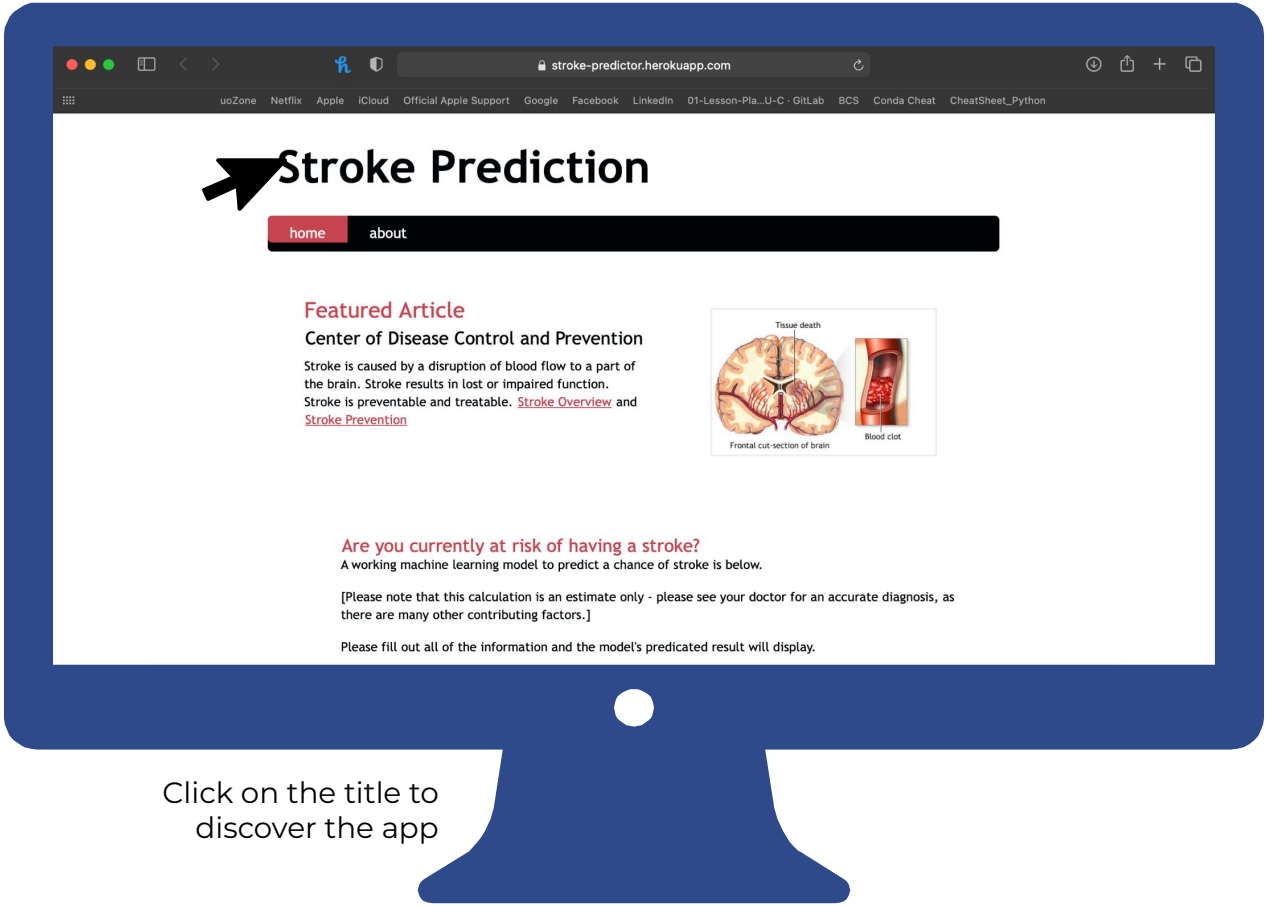
	precision	recall	f1-score	support
0	0.96	0.93	0.95	968
1	0.93	0.96	0.95	967
accuracy			0.95	1935
macro avg	0.95	0.95	0.95	1935
weighted avg	0.95	0.95	0.95	1935

Post-Optimization

Accuracy: **95.0%**
Recall: **96.0%**
F1-Score: **95.0%**

05

Front End Demo



Click on the title to discover the app

07

Limitations & Future Considerations

Limitations & Future Considerations

01 Limited Model Types Tested

02 SMOTE Optimization Usage

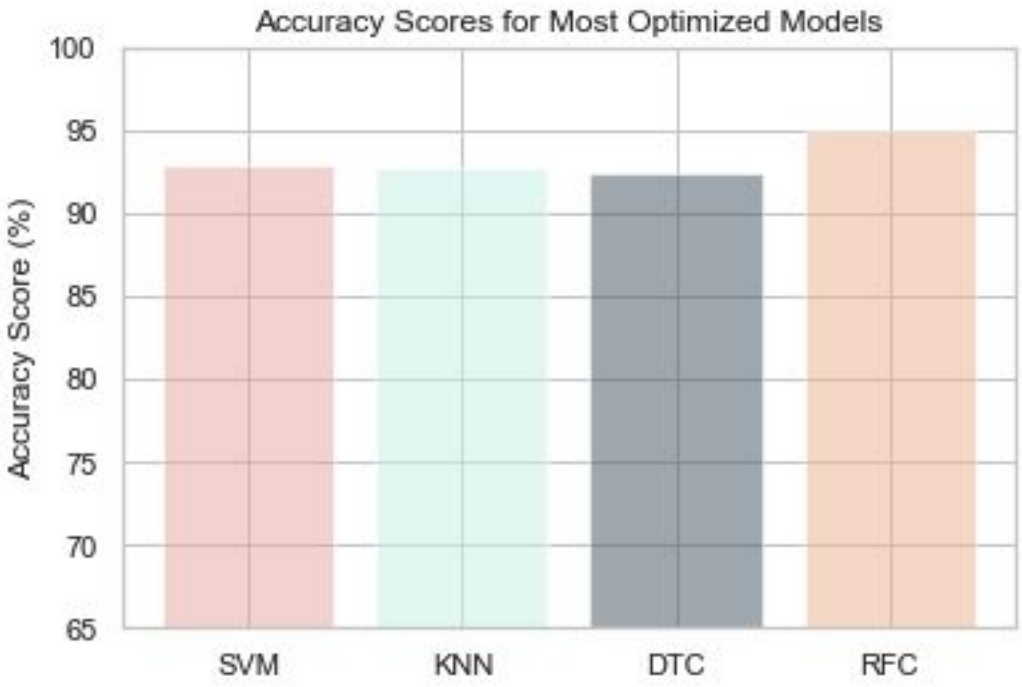
03 Stroke Data Features

04 Low Correlation Amongst Features

06

Conclusions

- All Models reached above 90% accuracy score
- RFC: 95%
- Best Choice:
  - Higher Accuracy
  - Recall
  - Individual Features of Input



08

Resources

Resources

Data set Link: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

(1) http://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html
(2) https://www.cdc.gov/stroke/about.htm
(3) https://avinetworks.com/glossary/anomaly-detection/
(4) https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
(5) https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee
(6) https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
(7) https://scikit-learn.org/stable/supervised_learning.html
(8) https://realpython.com/flask-by-example-part-1-project-setup/

Github Link of Flask App Heroku Link Github Link to Model Code