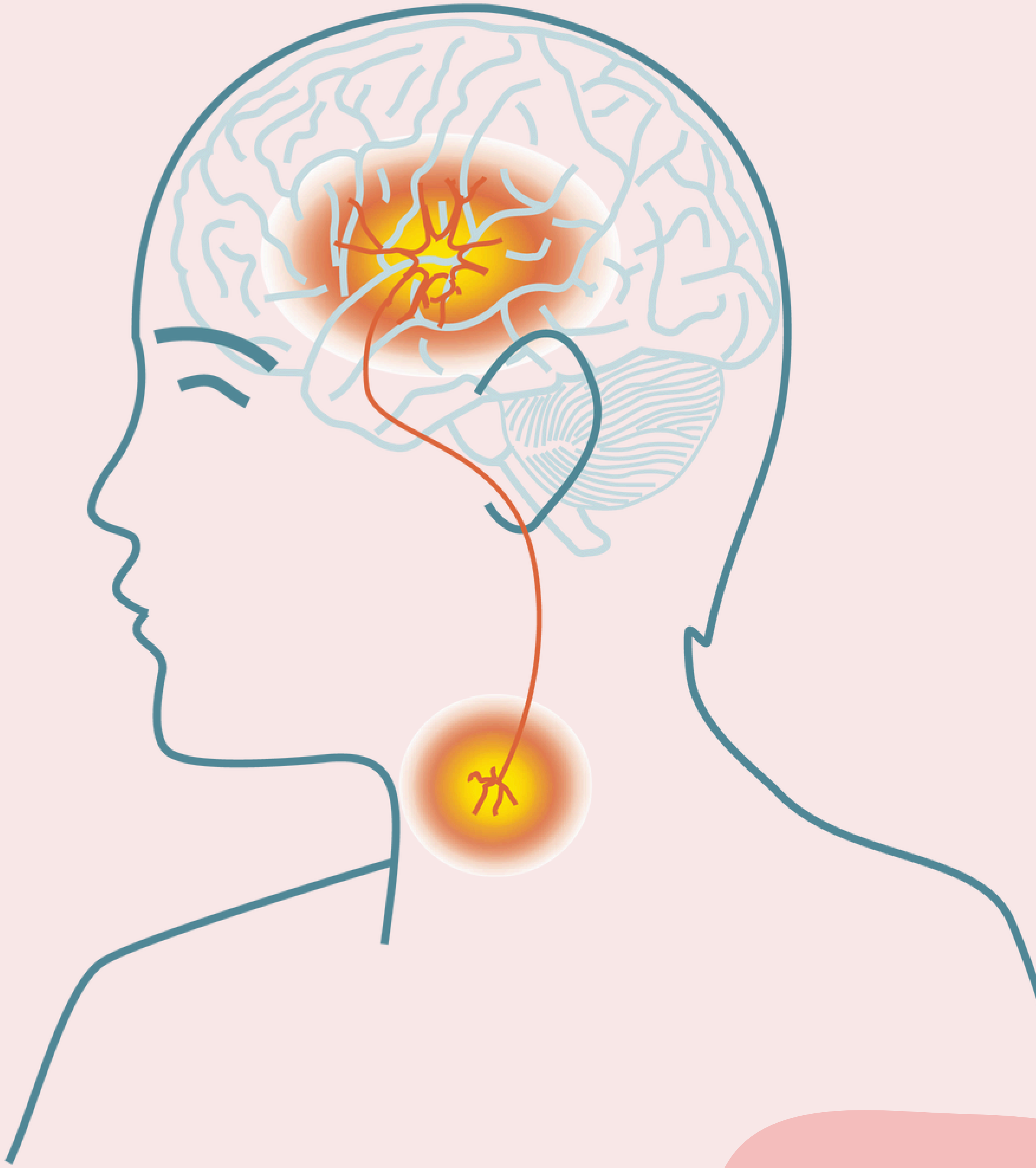
ML FOR STROKE PREDICTION

Group: “Daebak Data”



MOTIVATION + CODE

According to the National Institute of Child Health and Human Development (NICHD), “...about 795,000 people in the United States have strokes, and of these incidents, 137,000 of the people die.” Being able to predict whether not a patient is likely to get a stroke could help assist health professionals in identifying at risk patients, which could in turn lead to the adjustment of a patient’s treatment plan or promotion of stroke-training to their family members.

GitHub: https://github.com/atassiad/ML-for-Stroke-Prediction

https://www.nichd.nih.gov/health/topics/stroke/conditioninfo/risk

THE DATASET

The Kaggle stroke prediction dataset contains over 5 thousand samples with 11 total features (3 continuous) including age, BMI, average glucose level, and more. The output attribute is a binary column titled “stroke”, with 1 indicating the patient had a stroke, and 0 indicating they did not.

PRE-PROCESSING

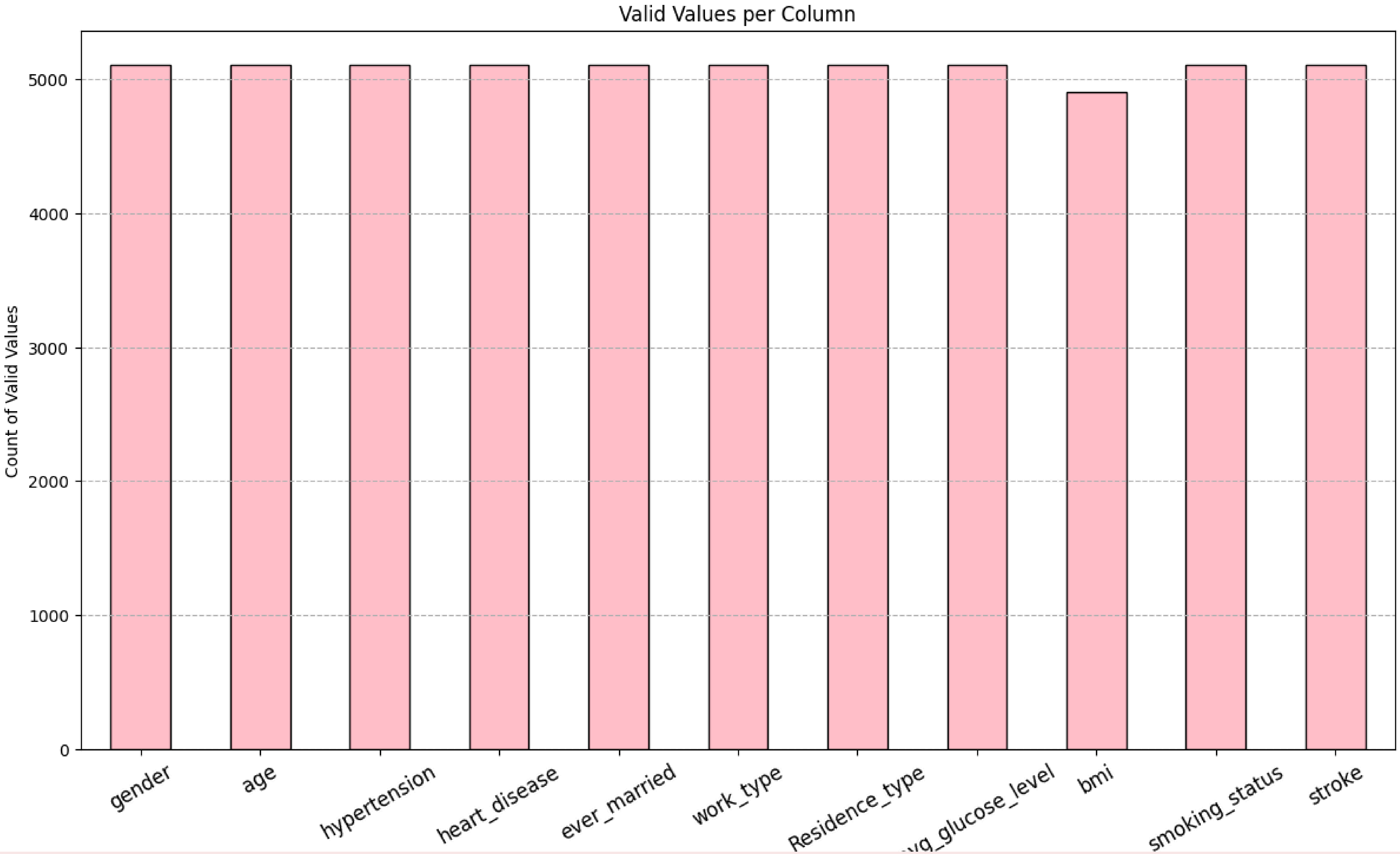
Problems:

1.) Irrelevant features (patient id)
2.) Null/NAN/Empty samples (BMI)
3.) Non-encoded categorical variables (marriage ,gender, residence, smoking status, work type))
4.) Imbalanced Dataset (95% no-stroke, only 5% stroke)

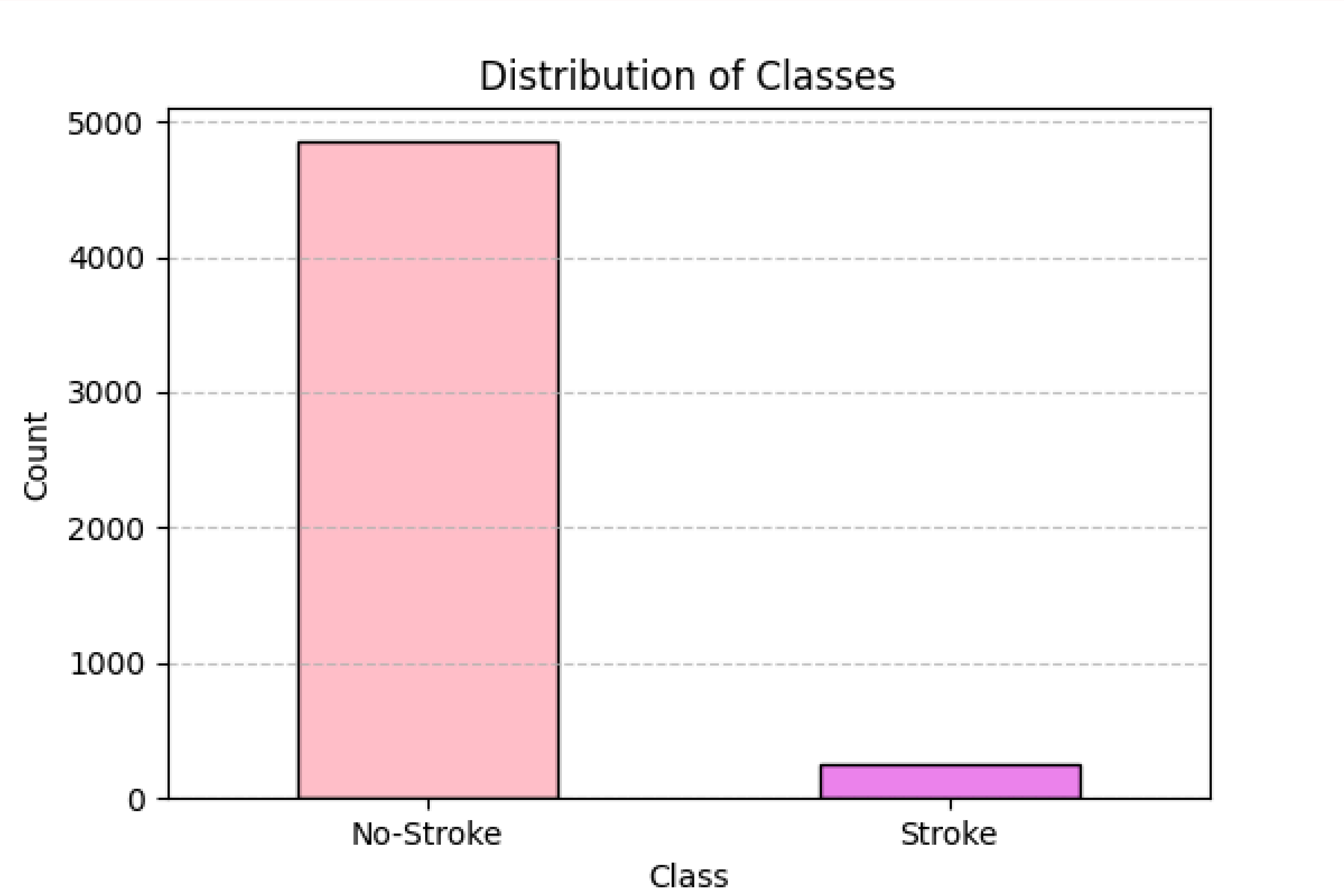
Solutions:

1.) Drop feature categories
2.) Fill with median values
3.) One-hot-encode categorical variables
4.) K-Fold Cross Validation + Under/Over Sampling

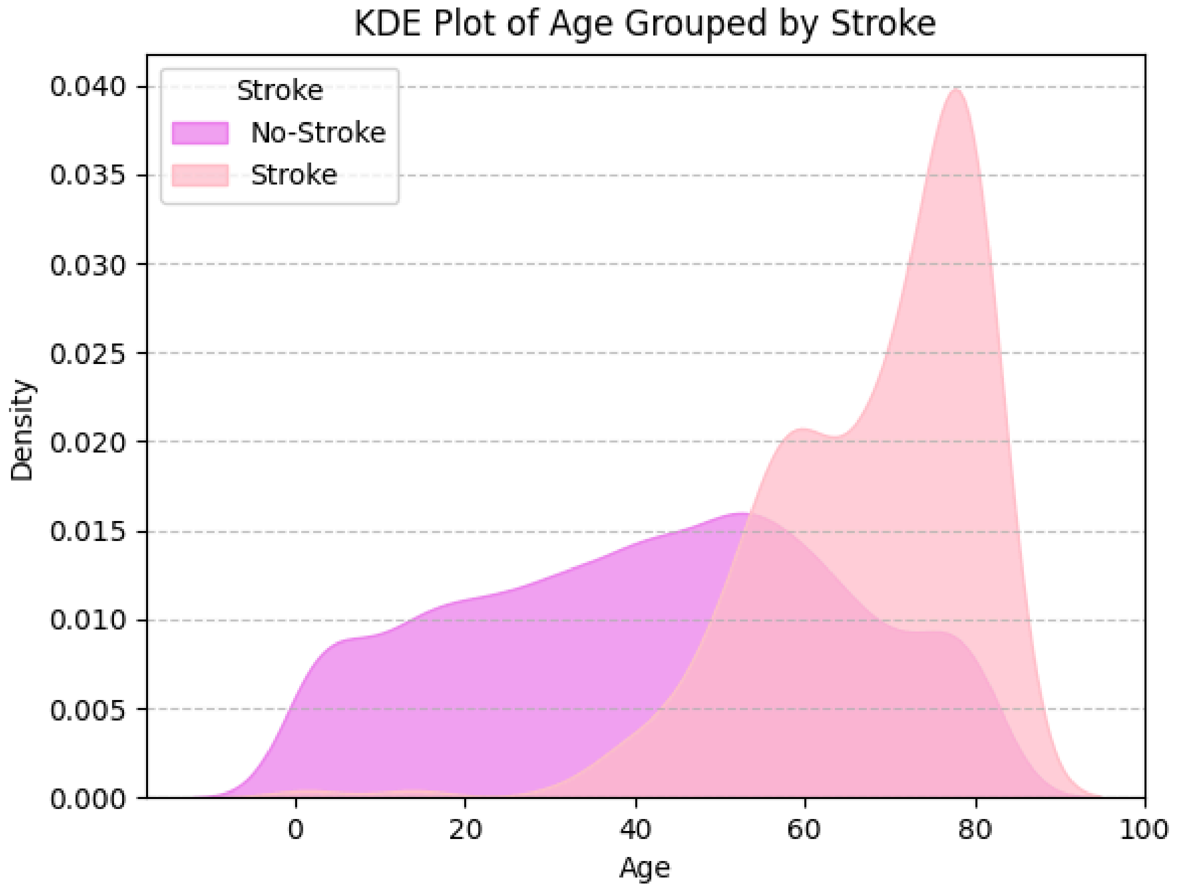
NULLITY-VISUALIZED

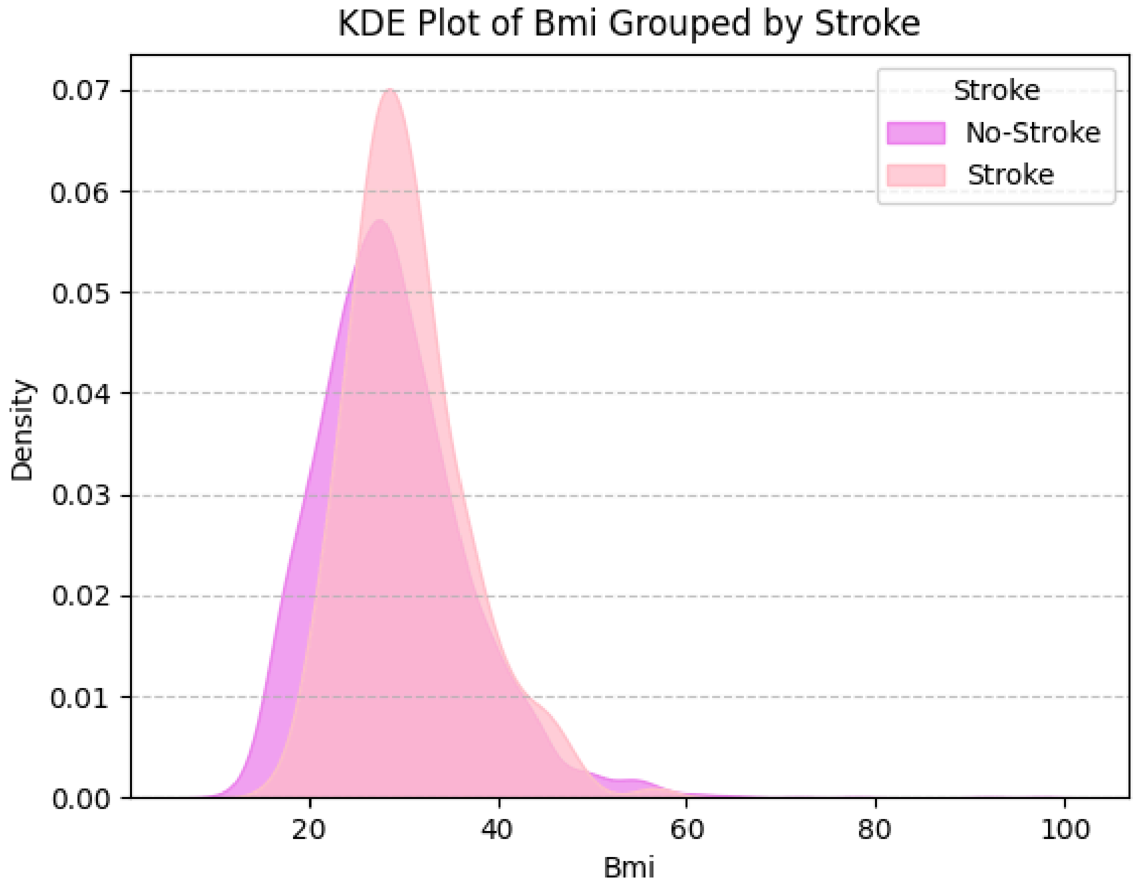


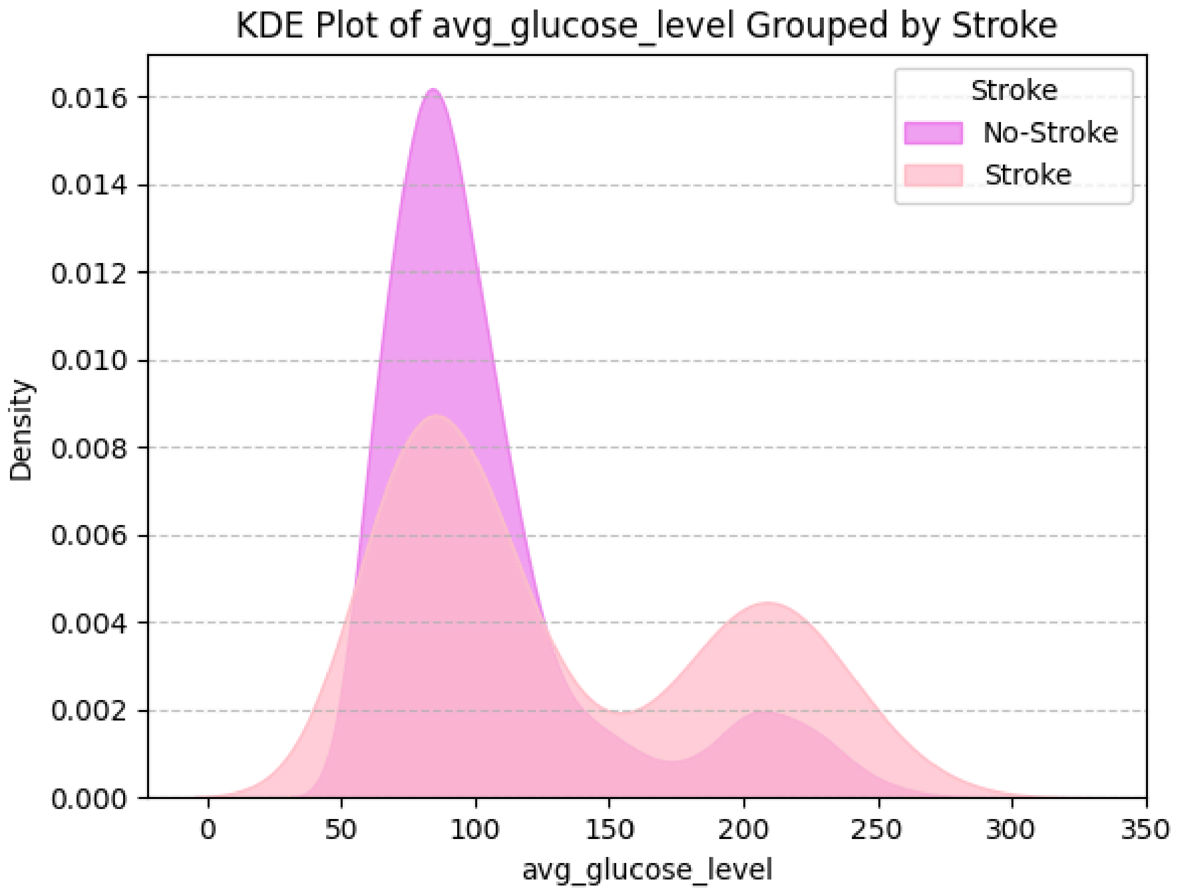
CLASS DISTRIBUTION



FEATURE RELVANCY (VISUALIZED)


KDE Plot of Age Grouped by Stroke


KDE Plot of Bmi Grouped by Stroke


KDE Plot of avg_glucose_level Grouped by Stroke

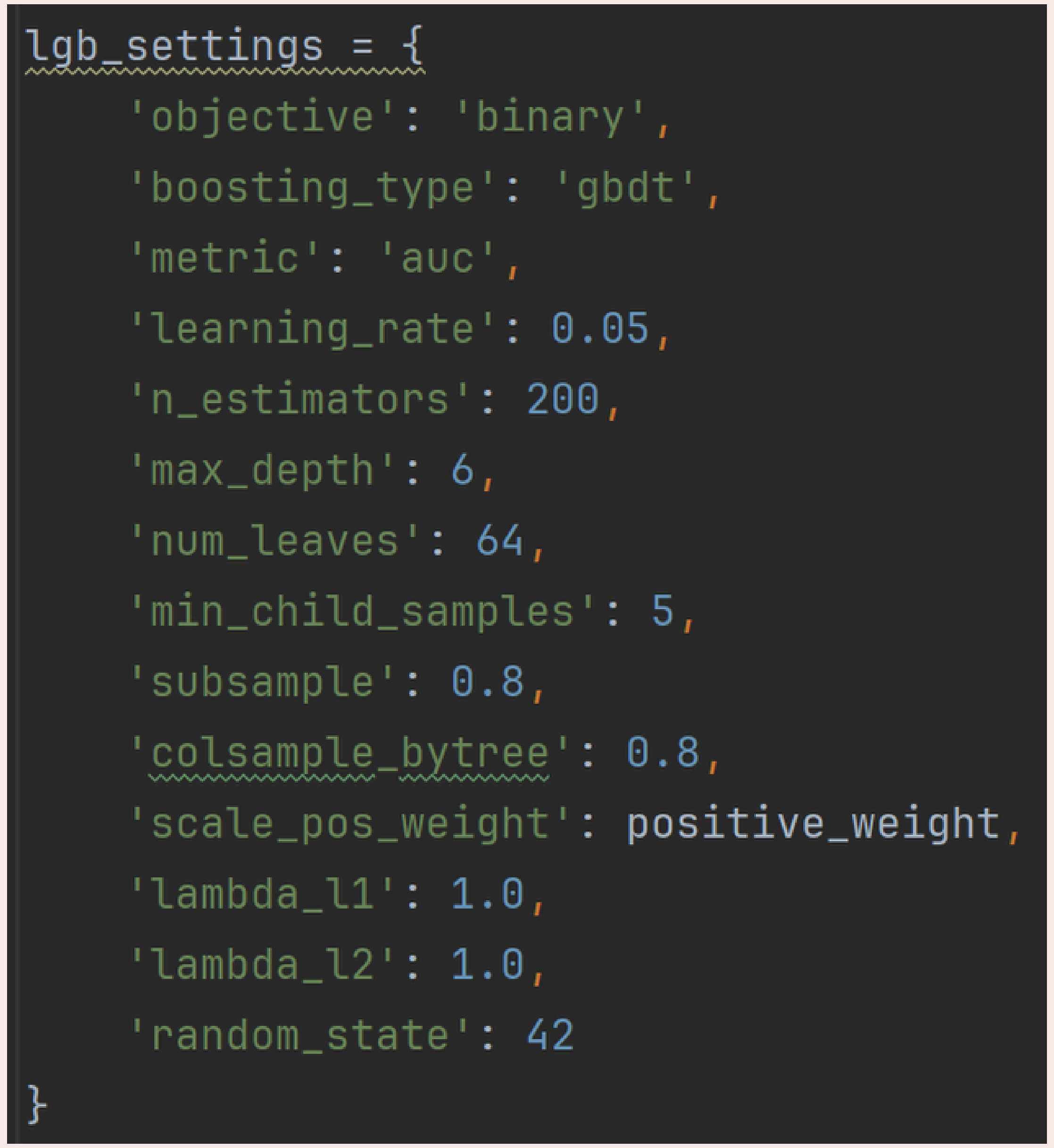
*Removing BMI & Glucose Level still resulted in similar measurements

PEFORMANCE

Model	Random Forest (Thanos)	Decision Tree (Thanos)	NN (Thanos)	KNN (John)	SVM (Keith)	GBM (Amy)
Precision	0.8015	0.7953	0.7244	0.7631	0.9321	0.1725
Recall	0.8595	0.9478	0.7713	0.8470	0.7671	0.6189
AUC	0.7933	0.7450	0.7288	0.7909	0.7802	0.8241

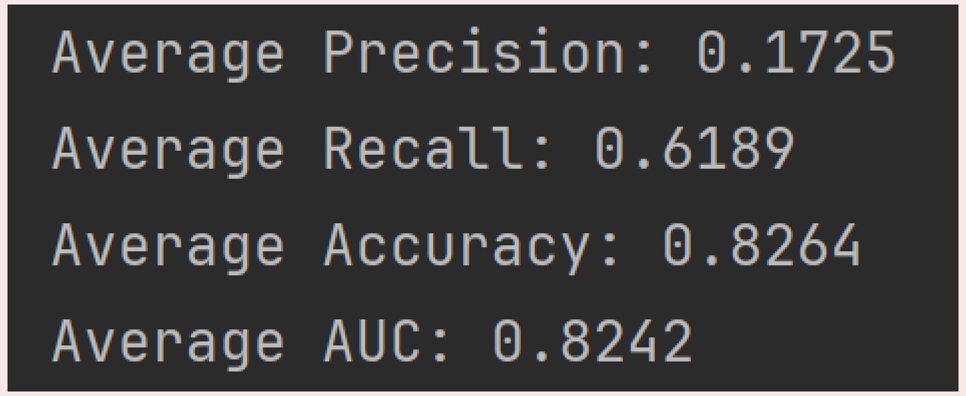
GRADIENT BOOST MODEL (GBM)

- Implemented LightGBM
- Dropped irrelant features: ID, work_type, residence_type, ever_married
- Normalized features using StandardScaler
- Undersampled and used k-fold cross validation to handle imbalance



```
lgb_settings = {
    'objective': 'binary',
    'boosting_type': 'gbdt',
    'metric': 'auc',
    'learning_rate': 0.05,
    'n_estimators': 200,
    'max_depth': 6,
    'num_leaves': 64,
    'min_child_samples': 5,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'scale_pos_weight': positive_weight,
    'lambda_l1': 1.0,
    'lambda_l2': 1.0,
    'random_state': 42
}
```

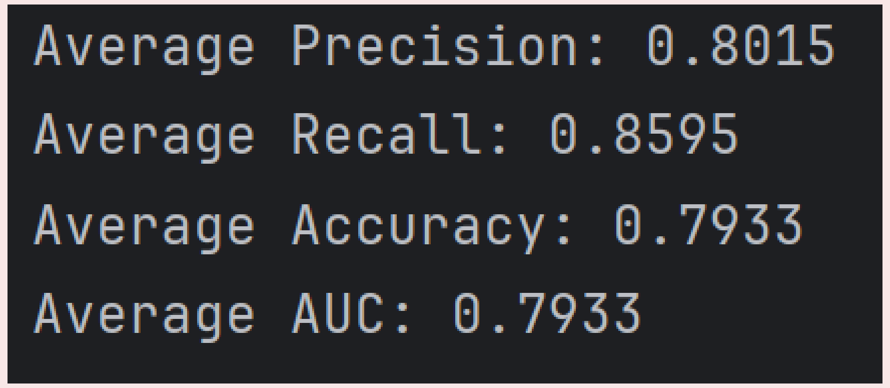
RESULTS:



```
Average Precision: 0.1725
Average Recall: 0.6189
Average Accuracy: 0.8264
Average AUC: 0.8242
```

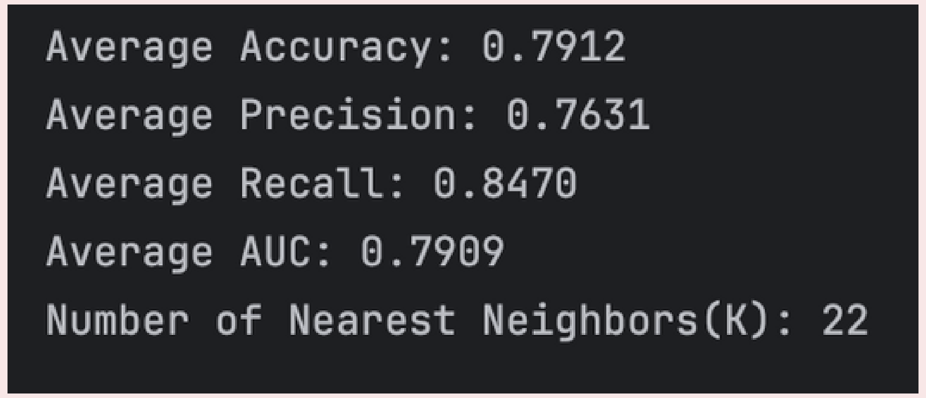
PROOF

RANDOM FOREST



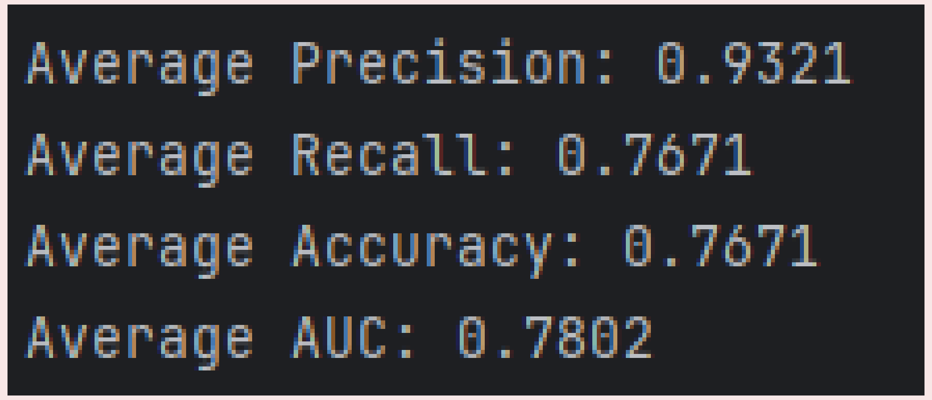
```
Average Precision: 0.8015
Average Recall: 0.8595
Average Accuracy: 0.7933
Average AUC: 0.7933
```

KNN



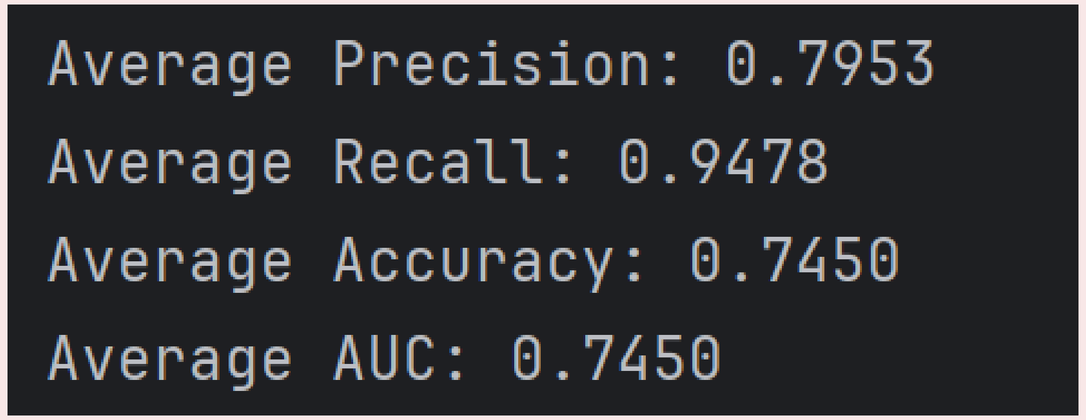
```
Average Accuracy: 0.7912
Average Precision: 0.7631
Average Recall: 0.8470
Average AUC: 0.7909
Number of Nearest Neighbors(K): 22
```

SVM



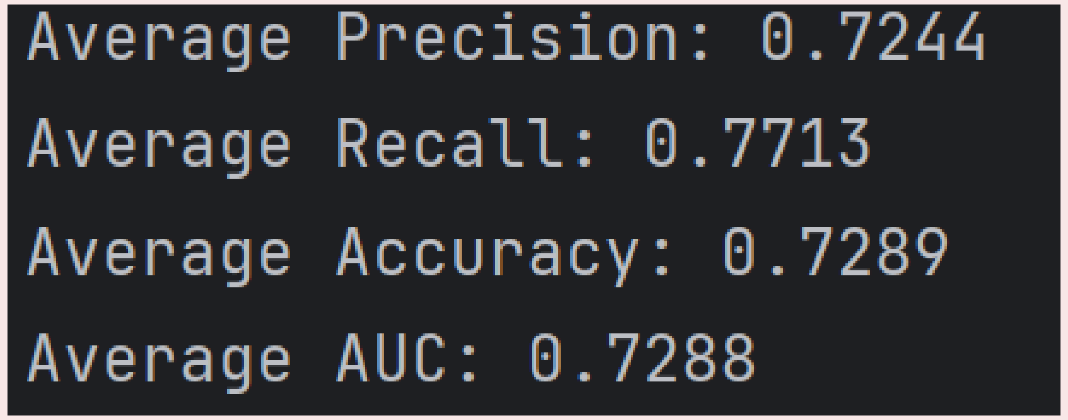
```
Average Precision: 0.9321
Average Recall: 0.7671
Average Accuracy: 0.7671
Average AUC: 0.7802
```

DECISION TREE



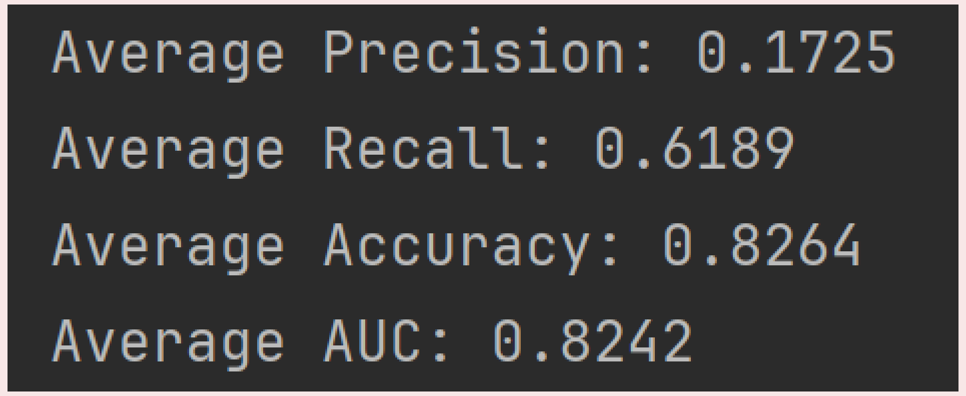
```
Average Precision: 0.7953
Average Recall: 0.9478
Average Accuracy: 0.7450
Average AUC: 0.7450
```

NN



```
Average Precision: 0.7244
Average Recall: 0.7713
Average Accuracy: 0.7289
Average AUC: 0.7288
```

GBM

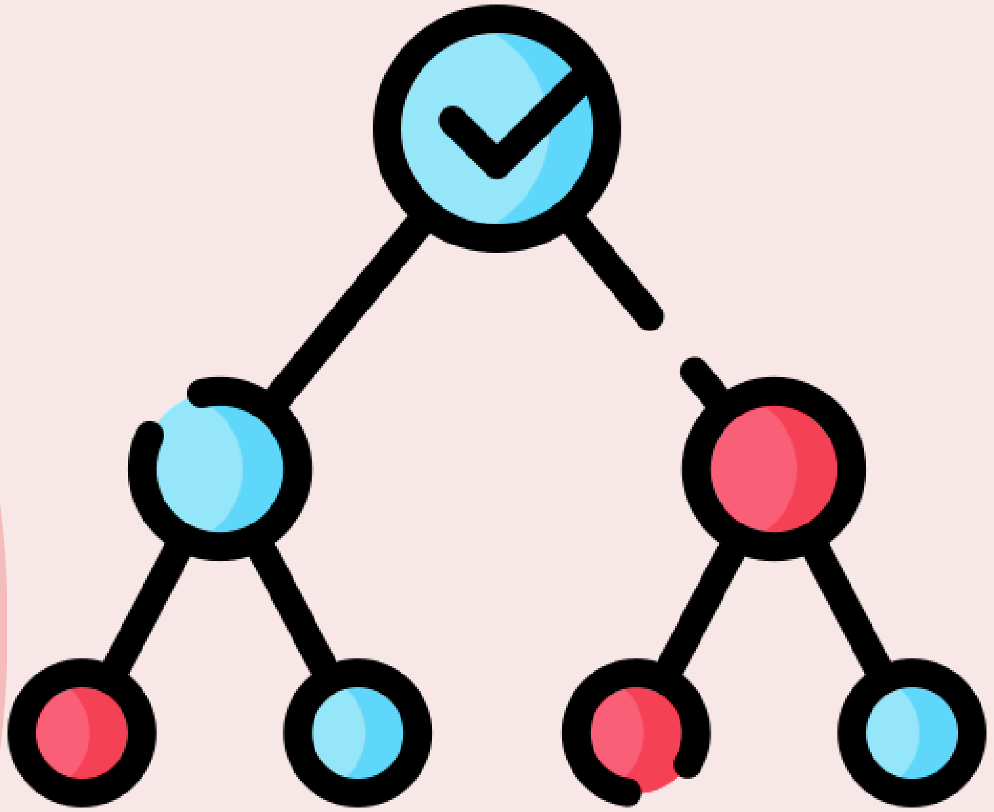


```
Average Precision: 0.1725
Average Recall: 0.6189
Average Accuracy: 0.8264
Average AUC: 0.8242
```

OPTIMAL PARAMETERS

Decision Tree / Random Forest

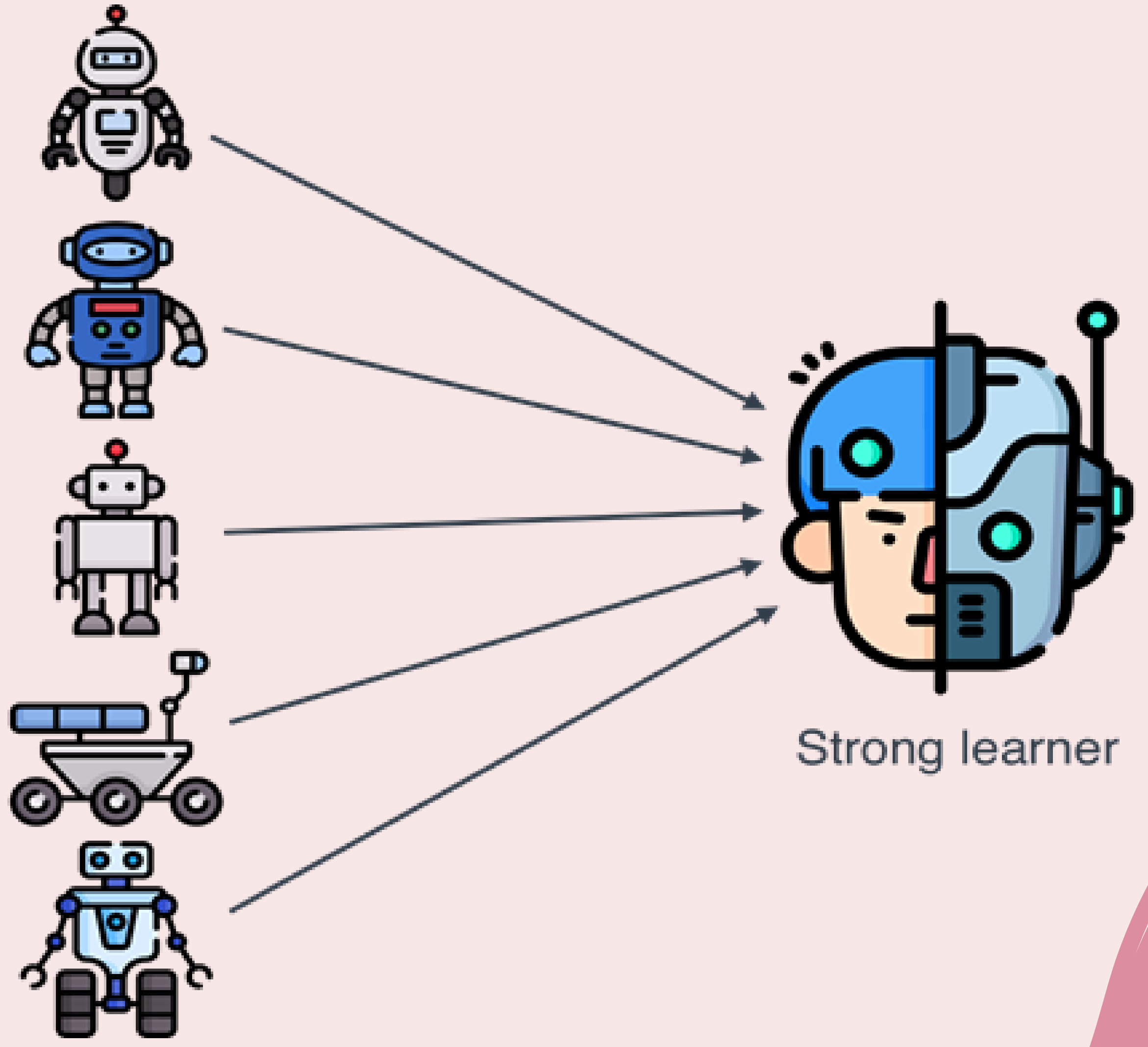
- Found that keeping the max depth small (around 2ish), prevented the model from overfitting to the majority class which led to higher recall.
- Entropy was most accurate measure of impurity (GINI was a close second)
- N_Estimator’s/# Of Trees: 50-250 estimators generally produced the same results. Less estimators increased the recall, likely due to overfitting to the majority class with too many.



FUTURE CHANGES / THINGS TO CONSIDER

Ensemble Learning: All our models capped off around the 75% - 80% accuracy range due to the class imbalance. Introducing ensemble techniques may have allowed for error correction on misclassified classes (mostly stroke) which could have led to a better overall performance (maybe decision tree for stroke class + another model)

Synthetic Minority Oversampling Technique: Introducing SMOTE may have improved the model’s ability to capture the minority class (stroke).



THANK YOU