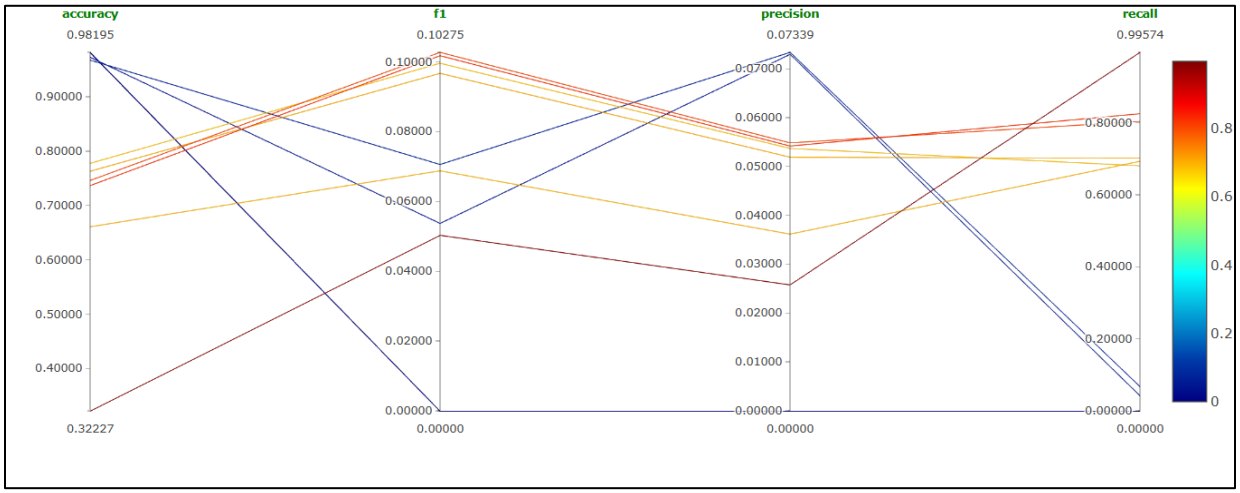
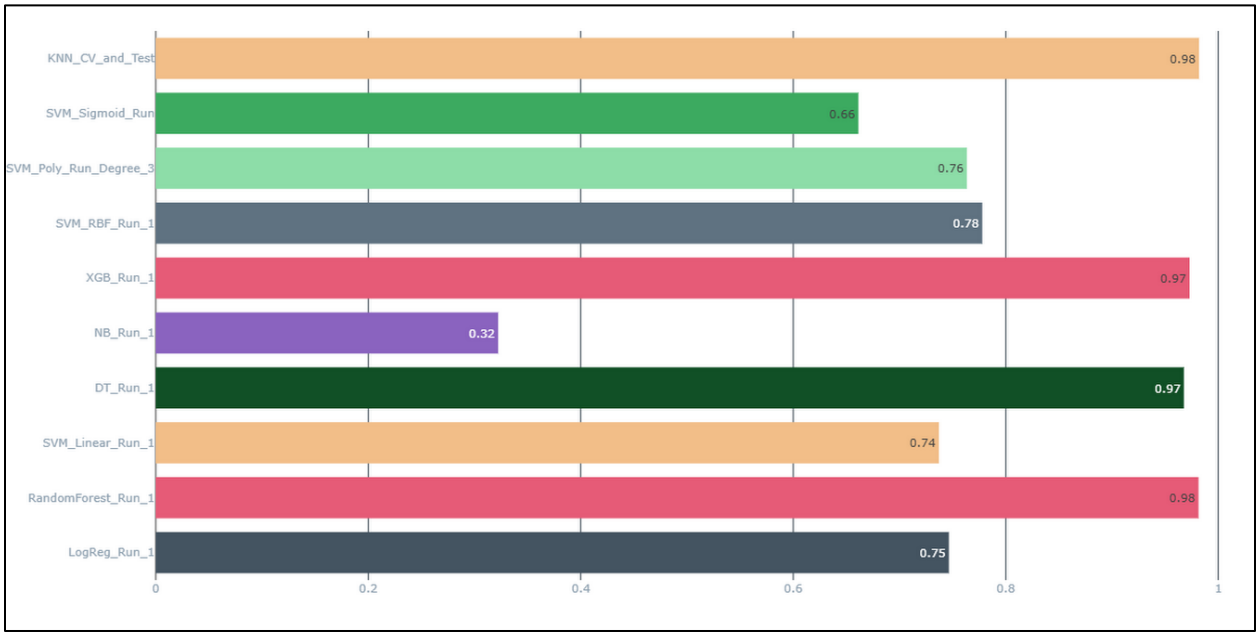
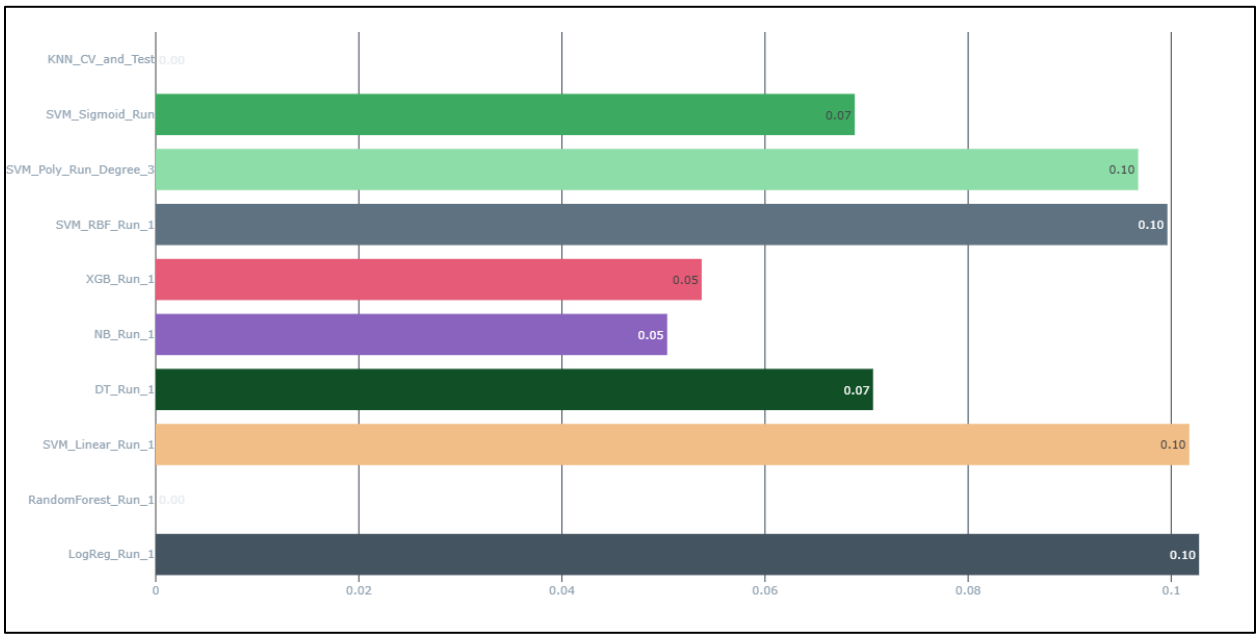
Comparison of Results of different Models



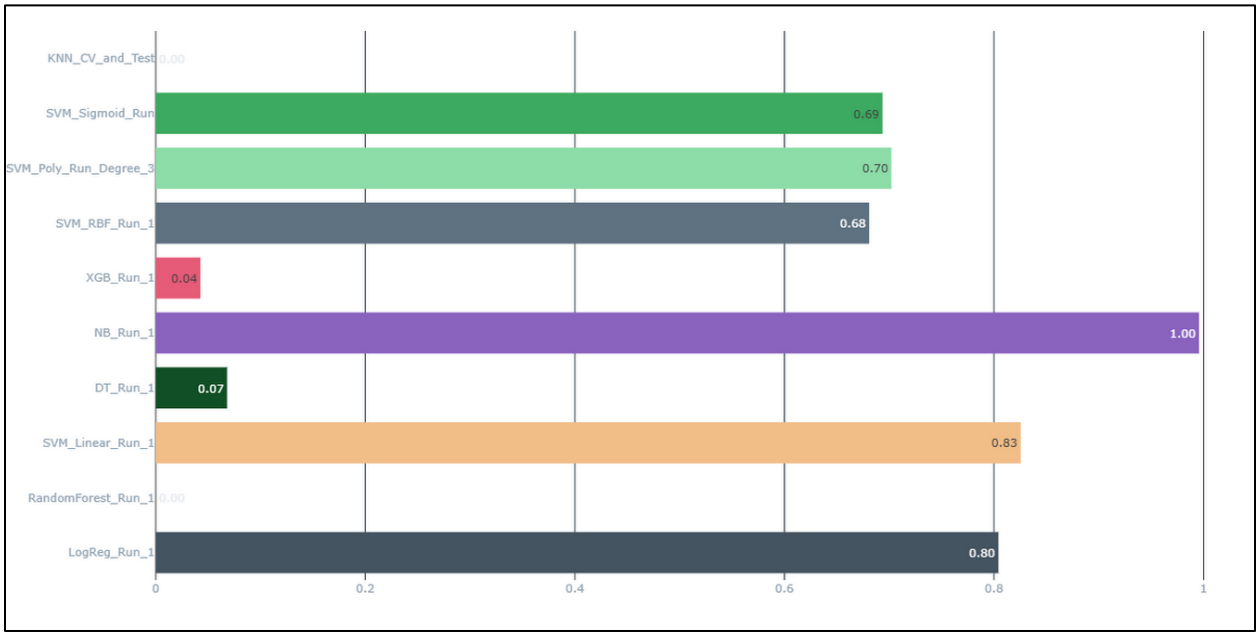
Accuracy Score of different Models



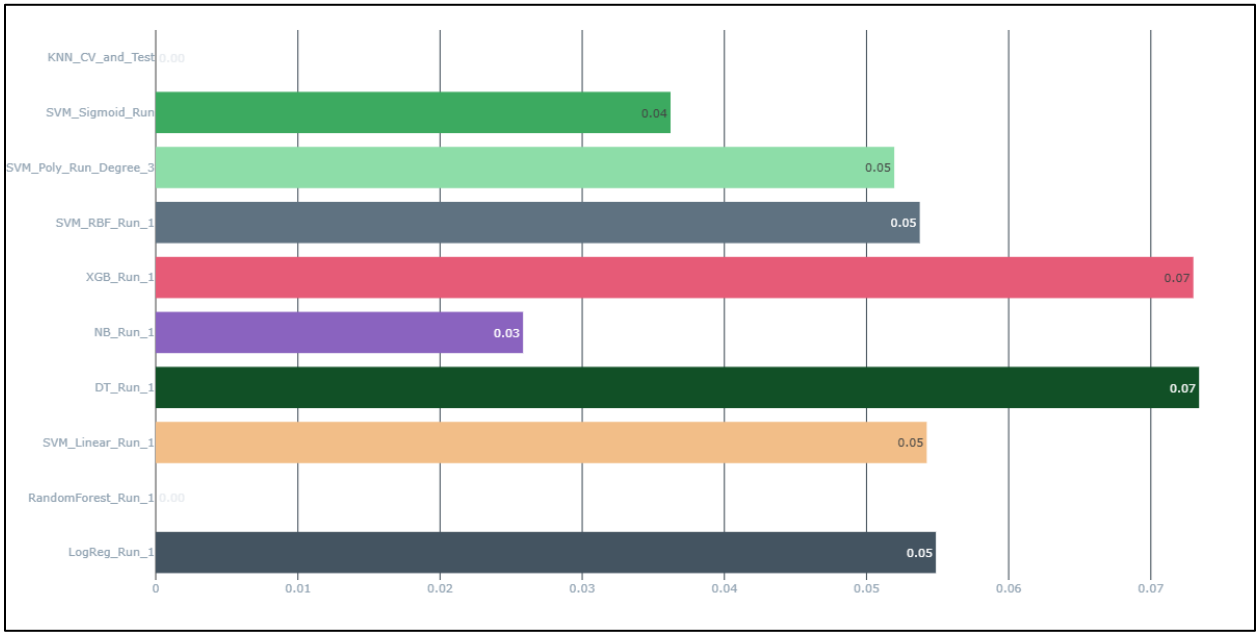
F1 Score of different Models



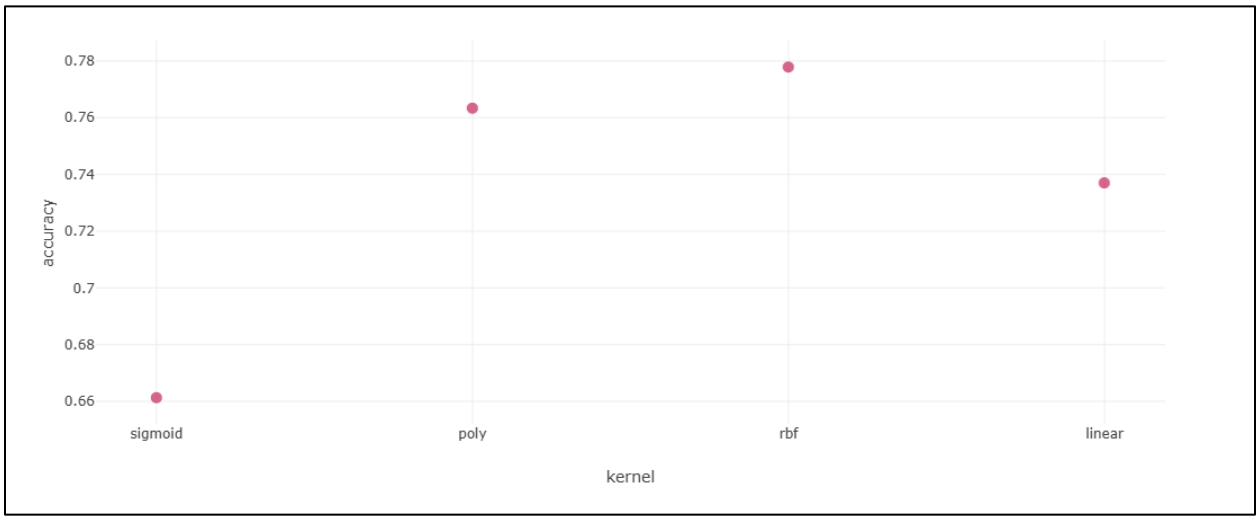
Recall score of different models



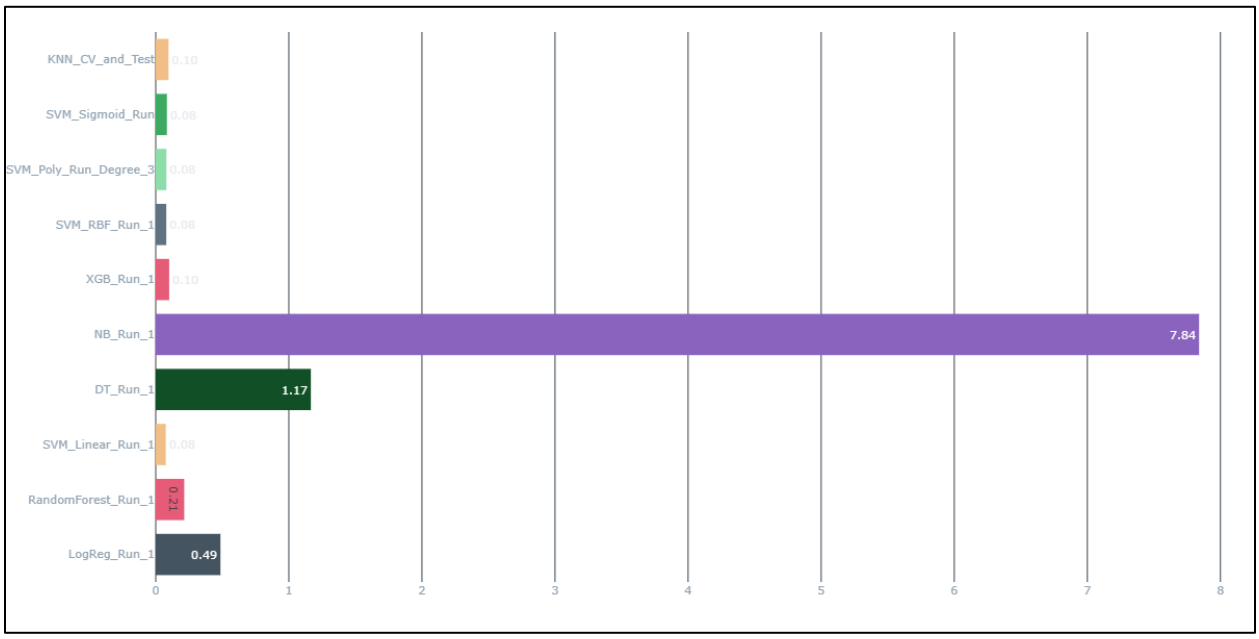
Precision Score of different models



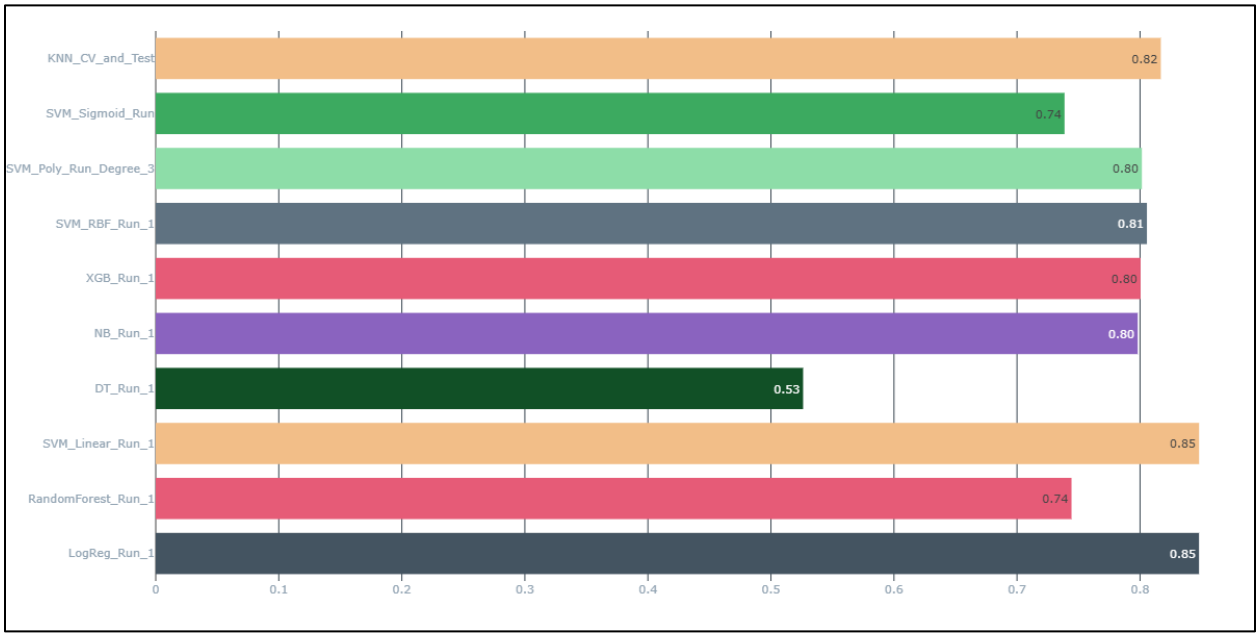
Accuracy Score of different SVM Kernels



Log Loss of different Models



ROC-AUC



Comparative Analysis: [2]

- **Compare the performance of your implemented model with the results reported in the selected paper.**

A. Comparison of results for SVM (with four kernels: sigmoid, polynomial, RBF, and linear) and KNN against the paper's findings.

Reported Metrics on paper

Kernel Type	Accuracy	Log Loss	ROC-AUC Score
Linear	0.9847	0.07856	0.6811
RBF	0.9847	0.07938	0.6274
Polynomial	0.9847	0.08556	0.5705
Sigmoid	0.9684	0.0771	0.5931

Achieved Metrics

Kernel Type	Accuracy	F1 Score	Log Loss	Precision	Recall	ROC-AUC
Sigmoid	0.661	0.069	0.084	0.036	0.694	0.739
Polynomial	0.763	0.097	0.08	0.052	0.702	0.801
RBF	0.778	0.1	0.079	0.054	0.681	0.806
Linear	0.737	0.102	0.075	0.054	0.826	0.848

The paper reports extremely high accuracy (>98%) across all SVM kernels , while our implementation yielded lower accuracy values (<0.8).

Based on the comparative analysis of implemented models and the results reported in the selected paper, it is evident that there are both similarities and discrepancies in model performance.

The paper reports extremely high accuracy scores for SVM models (>98% across all kernels), with the linear kernel achieving the best ROC-AUC score of 0.6811, followed by RBF (0.6274), sigmoid (0.5931), and polynomial (0.5705). In contrast, our implementation yielded lower accuracy values (linear: 0.737, RBF: 0.778, polynomial: 0.763, sigmoid: 0.661) but significantly higher ROC-AUC scores (linear: 0.848, RBF: 0.806, polynomial: 0.801, sigmoid: 0.739), indicating better class separation ability despite lower classification accuracy.

Our KNN model, using k=173 and Euclidean distance, achieved a high accuracy of 0.982 and ROC-AUC of 0.817, closely aligning with the paper's findings (accuracy ~0.98, ROC-AUC 0.8218).

Additionally, we implemented other models such as Logistic Regression, Random Forest, XGBoost, and Naive Bayes, which provided valuable insights into alternative approaches for stroke prediction.

Logistic Regression showed balanced performance with an ROC-AUC of 0.8479 and recall of 0.8043, while Random Forest and XGBoost demonstrated high accuracy but poor recall, highlighting issues related to class imbalance. Overall, our models performed well in terms of ROC-AUC and log loss, but some exhibited limitations in precision and recall due to the imbalanced nature of the dataset. These differences may stem from variations in train-test split strategies, lack of class imbalance handling techniques like SMOTE, and differences in feature engineering or hyperparameter tuning.

Discuss any differences or discrepancies observed between your implementation and the paper's results.

B. Dataset Mismatch (Possibility)

The dataset was downloaded from the author's GitHub as link was provided in the paper. The author mentioned 70-30% train-test split that was implemented. But confusion matrix numbers for 30% test dataset were much different from the papers. It means that dataset was not 100% the same as there might be some changes in dataset (after publication of the paper).

C. Dataset Splitting and Seed Value

One major difference lies in the random seed used for train-test splitting . The paper does not specify the random state, which means their training and testing splits may differ from ours. This variability can significantly impact model performance, particularly in imbalanced datasets like this one.

D. Class Imbalance Handling

Another critical factor contributing to the discrepancy is the lack of class imbalance handling techniques in both our implementation and the original paper. Since the dataset contains far fewer stroke cases than non-stroke cases, models tend to become biased toward the majority class. This leads to high accuracy but poor recall, as seen in several of our models (e.g., Random Forest, XGBoost).

To improve performance, future work should incorporate techniques such as:

- SMOTE oversampling
- Class weighting
- Threshold tuning

E. Feature Engineering and Preprocessing

The paper emphasizes feature engineering, including normalization and encoding of categorical features. Our implementation followed similar preprocessing steps (e.g., StandardScaler, label encoding), ensuring comparability. However, the paper also mentions factor analysis , which we did not perform. This might explain some of the variation in model behavior.

F. Model Complexity and Overfitting

The paper reports extremely high accuracy for SVM models, which may suggest overfitting. In contrast, our SVM models show more moderate accuracy but improved generalization (higher ROC-AUC), indicating a potentially healthier trade-off between bias and variance.

G. Evaluation Metrics

The paper focuses heavily on accuracy , which is misleading for imbalanced datasets. Our evaluation includes ROC-AUC , log loss , and other relevant metrics, offering a more comprehensive view of model performance.

H. Assumptions and unknown information

Several assumptions about model parameters and dataset were found and several information was missing for perfect implementation.