

Heart Stroke Prediction

Shuai Xu (sxu75374@usc.edu)

02/13/2022

1. Introduction

1.1. Approach Overview

Project Pipeline: Load Data -Check the information of the Dataset (missing value and categorical features, imbalance) - Data Scrubbing (with median value of bmi) – Data preprocessing (one hot encoding and Label encoding) – Data Visualization(corr) – Training data split into train set and validation set – Normalization train, val, and test based on the training set - Check the Feature importance – oversampling/undersampling – Choose evaluation metrics and model selection (8 models) – Use GridSearchCV to tune the hyperparameters of selected models (4 models)– change the threshold of prediction to get the final validation results.

2. Implementation

2.1. Data Set

- **Data Info:** The Heart Stroke dataset has 11 features and 1 binary output. The features include 4 integers, 2 float, and 5 categorical features. Training set has 3859 datapoints and Test set has 1251 datapoints.
- **Missing Values:** We could find that there are 150 missing values in the Training set and 51 missing values in the Test set.
- **Imbalance:** The Training set has 3676 non-stroke (95.2578%) and 183 stroke (4.7422%) datapoints, which the classes are totally imbalance.

2.2. Preprocessing Pipeline

2.2.1 Missing Value Imputation

In Training set, the missing value of 'bmi' takes 3.887% and in Test set missing value of 'bmi' takes 4.077%. I decide to use the Median value of the 'bmi' score in the Training set to fill the NaN of 'bmi' in both Training and Test set.

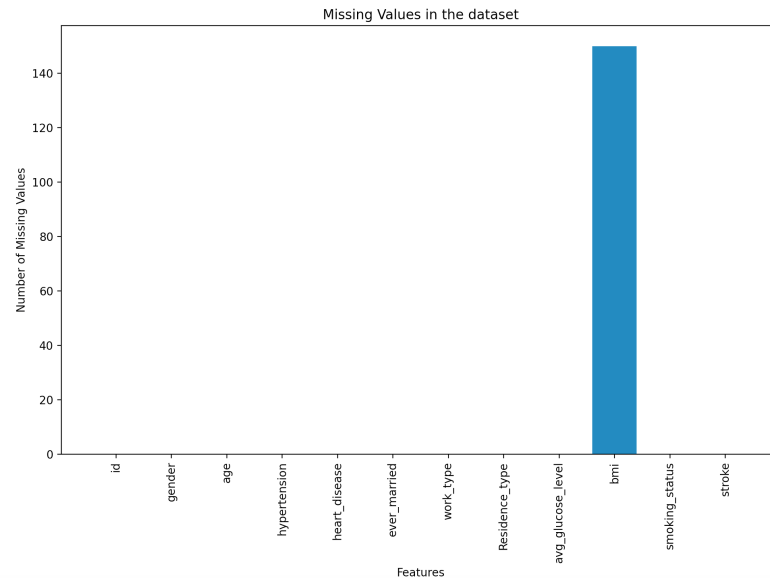


Figure 1: Missing Value

2.2.2 Categorical Feature Encoding

Table 1: categorical features

Features	Categories
gender	{'Male', 'Other', 'Female'}
ever_married	{'Yes', 'No'}
work_type	{'Never_worked', 'children', 'Private', 'Self-employed', 'Govt_job'}
Residence_type	{'Rural', 'Urban'}
smoking_status	{'never smoked', 'smokes', 'Unknown', 'formerly smoked'}

There are 5 categorical features, I use the label encoding to convert it into the label. For the 'Unknown' category in the smoking status, I deal it as an attribute in this project.

2.2.3 Data Visualization

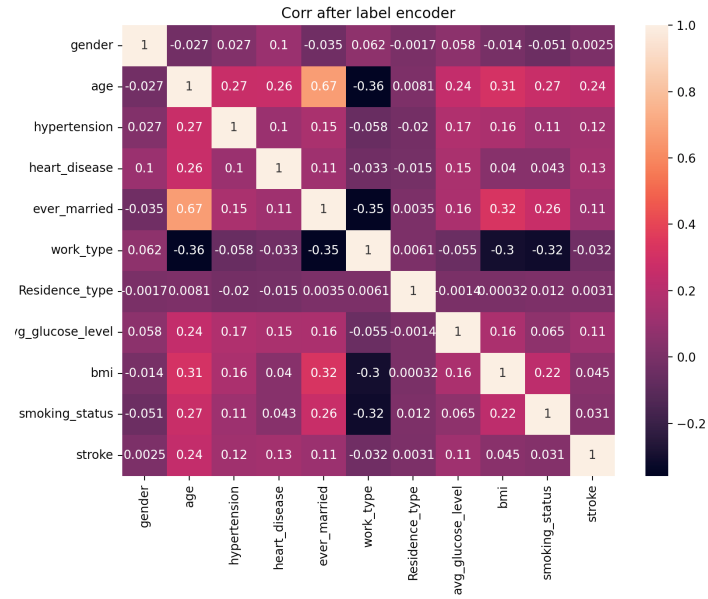


Figure 2: Correlation Heatmap after feature label encoding

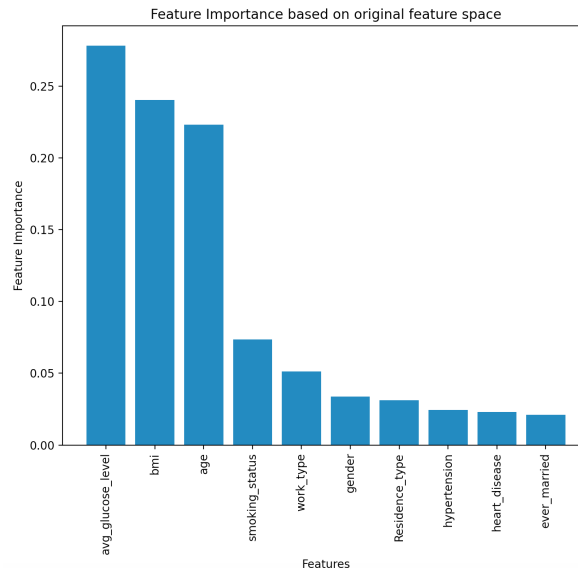


Figure 3: Feature Importance

Table 2: Feature Importance

Features	Feature importance
avg_glucose_level	0.278276
bmi	0.240369
age	0.223112
smoking_status	0.073565
work_type	0.051223
gender	0.033885
Residence_type	0.031126

hypertension	0.024334
heart_disease	0.023132
ever_married	0.02098

I use a default Random Forest classifier to get the feature importance score.

2.2.4 Data Usage

I firstly do the train-test split to separate the whole Training data (3859 datapoints) into the Train set (3087 datapoints) and Validation set (772 datapoints) with the ratio of 4:1 and only use the Test data for predicting the final output to prevent from data snooping.

2.2.5 Normalization

After splitting the Train set and Validation set, I do the normalization to rescale the dataset into the same range based on the Train set. Then, apply the coefficient of the Train set to the Validation set and Test set. The standardization method I chose is StandardScaler().

2.2.6 Dealing with Imbalance

After standardized, I use the oversample technique SMOTE to add some noise to create some datapoints for the class 'stroke=1' to deal with the datapoint imbalance. After oversampling, both of the class 'stroke=0' and 'stroke=1' have the same number of datapoints. We do oversample to eliminate the different class weights in an imbalanced dataset.

2.3. Training and Model Selection

2.3.1 Measurement Metrics

The final measurements I used: Accuracy, Precision, **Recall**, **Specificity**, **Sensitivity**, f1 score, **f2 score** (f beta score with beta=2) and **roc auc score**. Bold font techniques are more important in the measurement.

In the Heart Stroke dataset, two class is totally imbalanced and heart stroke datapoints will be easy to ignore to compare with the no heart stroke datapoints. Thus, I think we should focus on the Recall, Specificity, Sensitivity, f2 score and ROC AUC of the stroke data, but not just focus on the Accuracy, Precision and f1 score. A stroke prediction system needs to focus on the stroke detection, not a very high accuracy cause by only detecting no stroke datapoints. ROC AUC uses true positive rate and false positive rate as the y-axis and x-axis, that should be useful to ignore the imbalanced class weight and measure the performance in a general case. Also, Specificity and Sensitivity are not influenced by the true probability of the class label as objective measurement metrics. For f1 and f2 score, the high precision low recall and low precision high recall will give us the same f1 score, but we just need the high recall one, thus f1 score should not be

a good measurement for this dataset. F2 score has a larger beta to compare with the f1 score, it has a higher weight on recall to compare with f1.

2.3.2 Baseline

I set two baseline models, one trivial baseline and one non-trivial baseline.

- Trivial baseline: mode(). I use the mode of the Train set label as the y_pred which should be all 0 (no stroke) for our dataset.
- Non-trivial baseline: I use a 1-NN with default setting as the baseline model for the comparison.

2.3.3 Models

Firstly, I choose 8 models with default settings to basically select 4 of them by the cross-validation.

Table 3: model selection

Model	accuracy	precision	recall	f1_score	f2_score	roc_auc
Logistic Regression	0.72225	0.112203	0.7465	0.19497	0.34995	0.816098
Naive Bayes	0.702038	0.103397	0.7318	0.18112	0.33002	0.792201
Bernolli Bayes	0.552121	0.083915	0.902	0.15352	0.30566	0.765507
Decison Tree	0.892	0.115183	0.2088	0.14764	0.17869	0.566502
Random Forest Classifier	0.930936	0.119167	0.0834	0.09712	0.08827	0.796892
AdaBoost Classifier	0.844511	0.111453	0.3482	0.16846	0.24359	0.781387
Support Vector Machine	0.777845	0.087871	0.4199	0.14513	0.2386	0.735652
K Nearest Classifier	0.809909	0.082638	0.3195	0.1312	0.20283	0.610801

From the table above, we could see the Logistic Regression and Naïve Bayes have very good Recall, f2 score and ROC AUC score to compare with others. Also, I decide to choose SVM and KNN for the model selection, because they usually

2.3.4 Model Selection

- I use GridSearchCV to tune the hyperparameters for each of the models, the scoring method in CV I choose to use the ROC AUC as a measurement for general case.

The CV result shows below:

Table 4: CV result

CV result	Best CV score (roc auc)	Best settings
-----------	-------------------------	---------------

Logistic Regression	0.8284101754200108	'logisticregression__C': 0.01, 'logisticregression__penalty': 'l1', 'logisticregression__solver': 'liblinear', class_weight='balanced', max_iter=500,
Naive Bayes	0.8031420765027322	GaussianNB()
AdaBoost Classifier	0.7984480110175326	'adaboostclassifier__n_estimators': 50
Support Vector Machine	0.7534135141869717	'svc__degree': 2, 'svc__kernel': 'sigmoid', class_weight='balanced', probability=True
K Nearest Classifier	0.6479177676964274	'kneighborsclassifier__algorithm': 'auto', 'kneighborsclassifier__n_neighbors': 7

Then, using Validation set to test the model performance

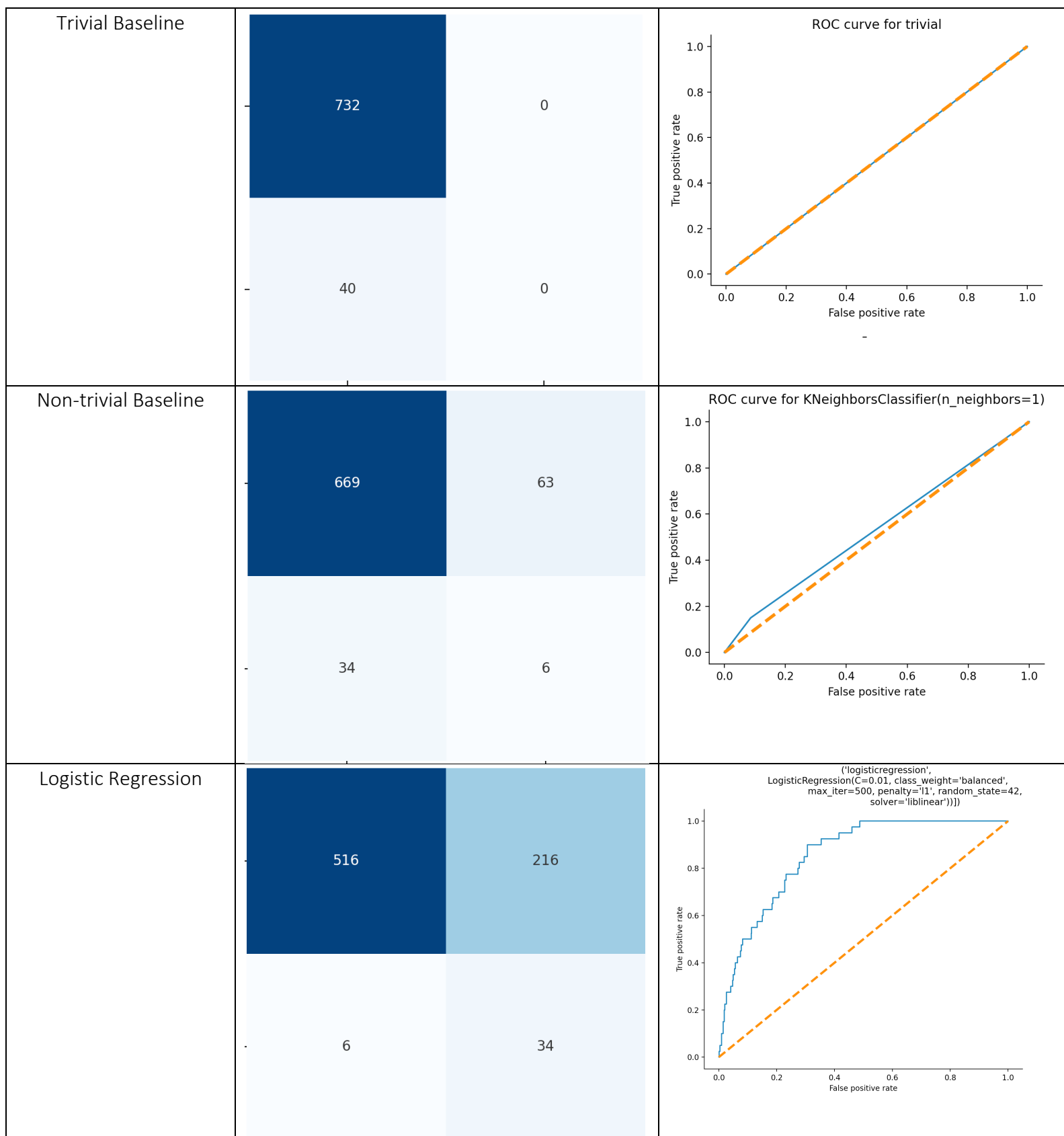
Table 5: Measurement Metrics for Validation set based on the best CV parameters

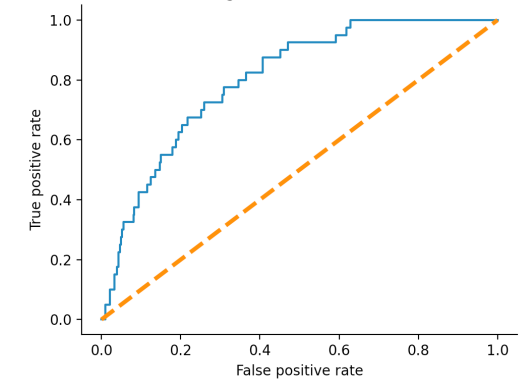
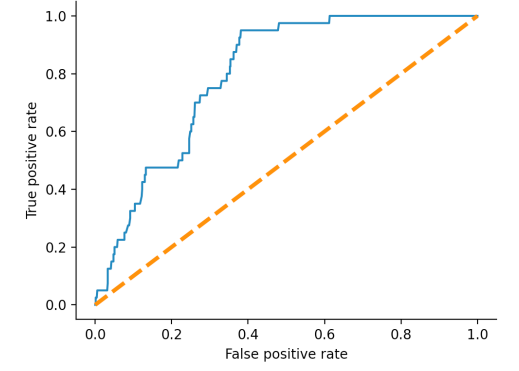
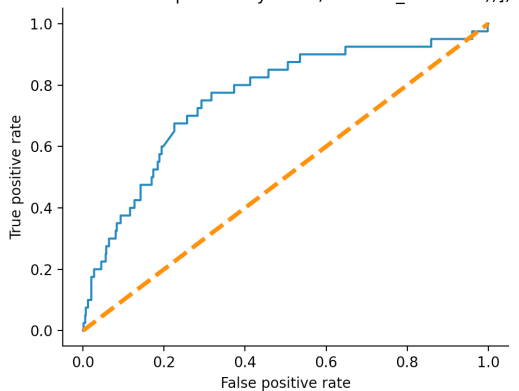
	accuracy	precision	recall	Specificity	Sensitivity	f1	f2	roc_auc
Trivial Baseline	0.95	0.0	0.0	1.0	0.0	0.0	0.0	0.49435
Non-trivial Baseline	0.87	0.08696	0.15	0.91393	0.15	0.11009	0.131	0.53197
Logistic Regression	0.71	0.136	0.85	0.70492	0.85	0.23448	0.41463	0.85451
Naive Bayes	0.7	0.11508	0.725	0.69536	0.725	0.19863	0.35194	0.80314
AdaBoost Classifier	0.85	0.14	0.35	0.88251	0.35	0.2	0.26923	0.7971
Support Vector Machine	0.79	0.14201	0.6	0.80191	0.6	0.22966	0.36474	0.76052
K Nearest Classifier	0.8	0.11184	0.425	0.81557	0.425	0.17708	0.27244	0.68248

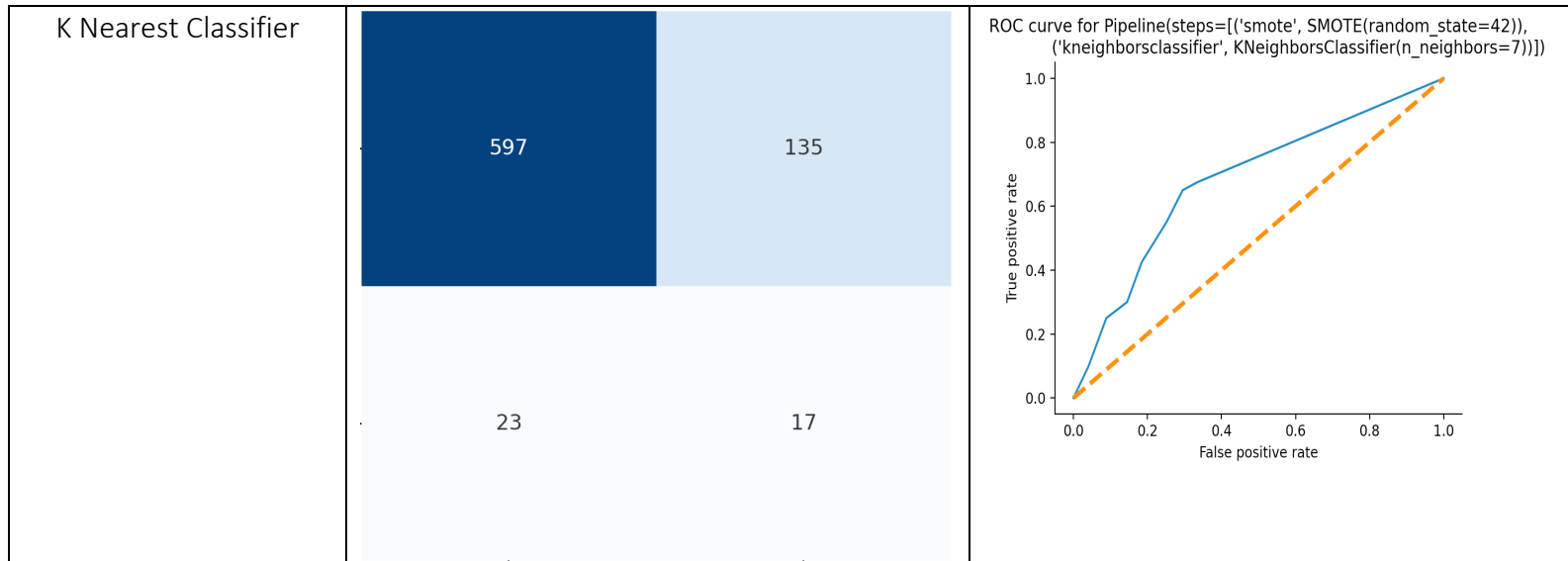
Confusion Matrix and ROC curve:

Table 6: Confusion Matrix and ROC curve for Validation set based on best CV parameters

	Confusion Matrix	ROC Curve
--	------------------	-----------



Naive Bayes	<table><tr><td>509</td><td>223</td></tr><tr><td>11</td><td>29</td></tr></table>	509	223	11	29	<p>ROC curve for Pipeline(steps=[('smote', SMOTE(random_state=42)), ('gaussiannb', GaussianNB())])</p> 
509	223					
11	29					
AdaBoost Classifier	<table><tr><td>646</td><td>86</td></tr><tr><td>26</td><td>14</td></tr></table>	646	86	26	14	<p>ROC curve for Pipeline(steps=[('smote', SMOTE(random_state=42)), ('adaboostclassifier', AdaBoostClassifier(random_state=42))])</p> 
646	86					
26	14					
Support Vector Machine	<table><tr><td>587</td><td>145</td></tr><tr><td>16</td><td>24</td></tr></table>	587	145	16	24	<p>SVC(class_weight='balanced', degree=2, kernel='sigmoid', probability=True, random_state=42))</p> 
587	145					
16	24					



2.3.5 Threshold Tuning: Final validation Result

Last, I tune the threshold of getting the prediction value from the probability model by using the best f1 score. If the predicted probability > the threshold we get, we consider it as 1, or we consider it as 0.

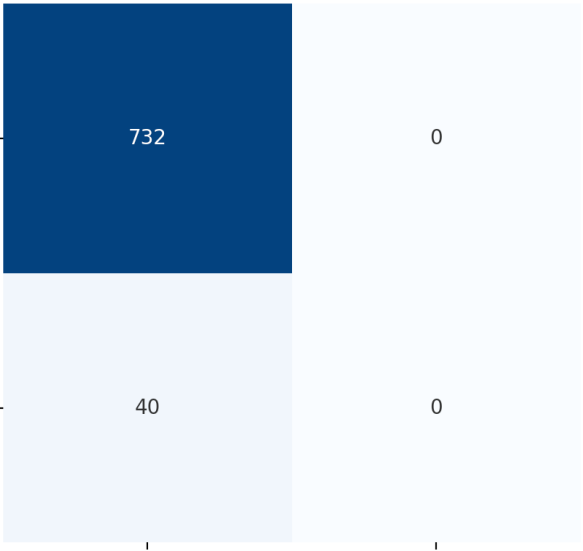
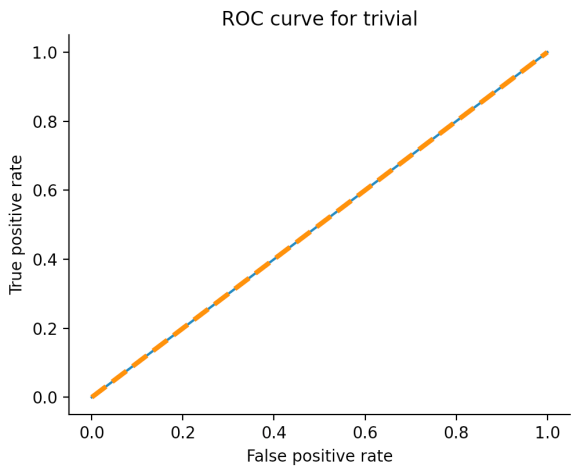
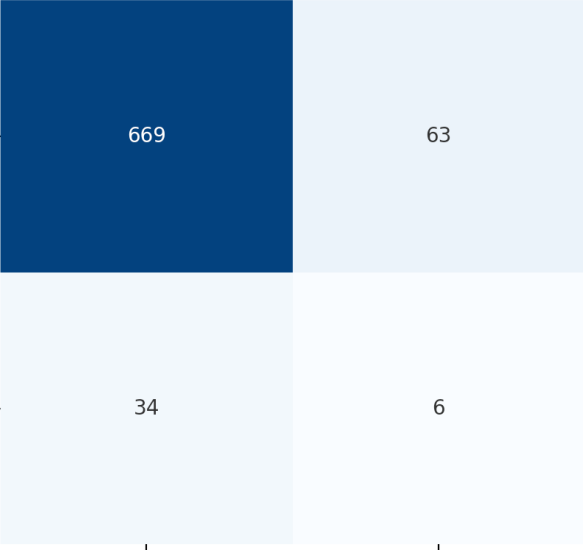
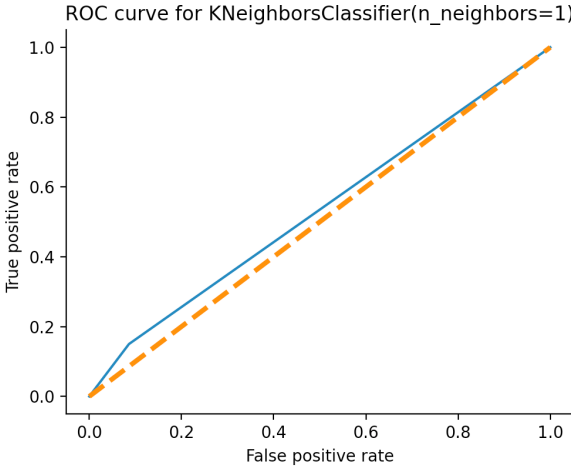
The threshold for each of the probability models I found based on the best f1 score is **SVC:0.57, Logistic Regression:0.59, Naïve Bayes: 0.59, KNN: 0.2, and AdaBoost: 0.49.**

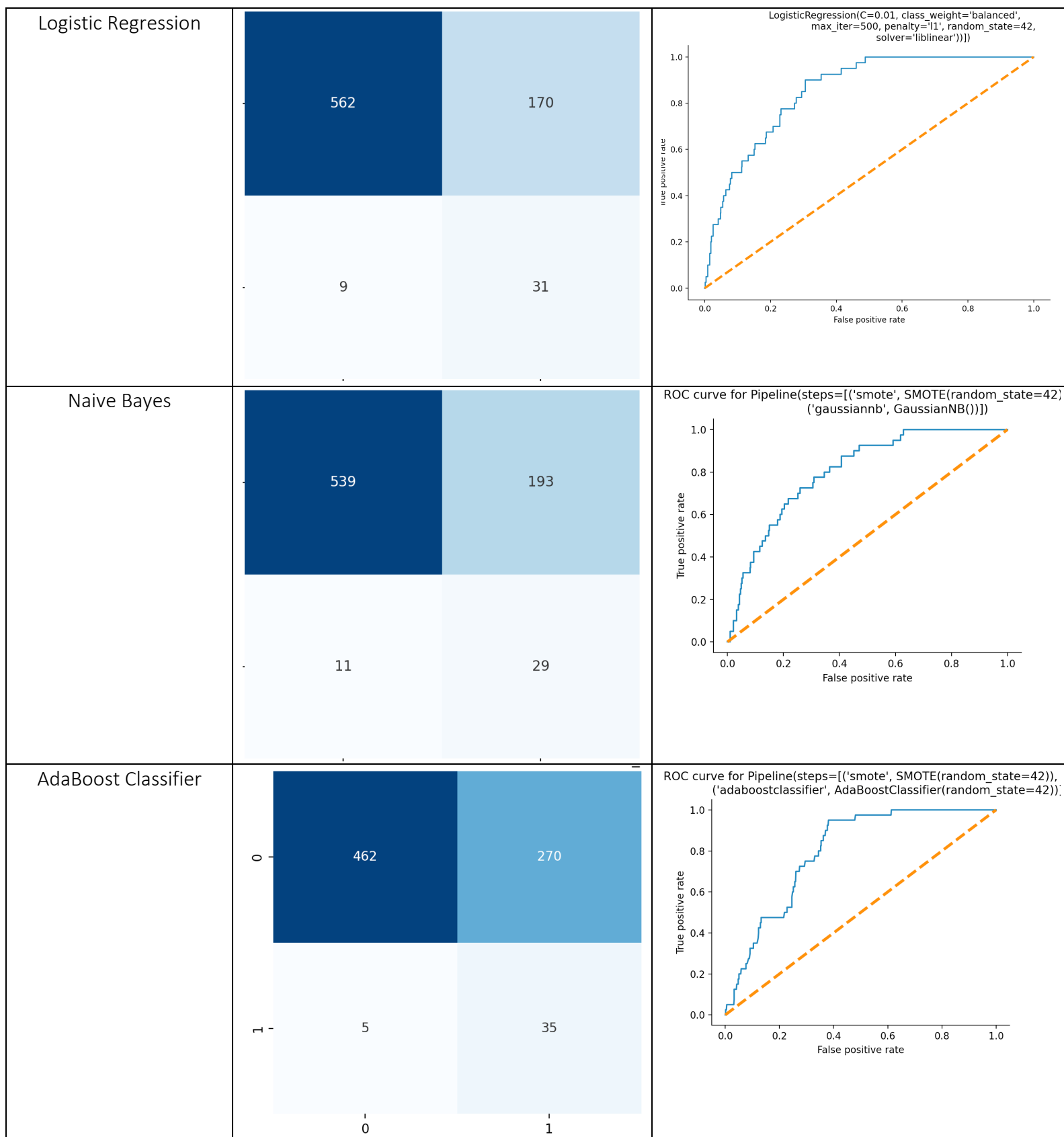
The following result after change the threshold shows below:

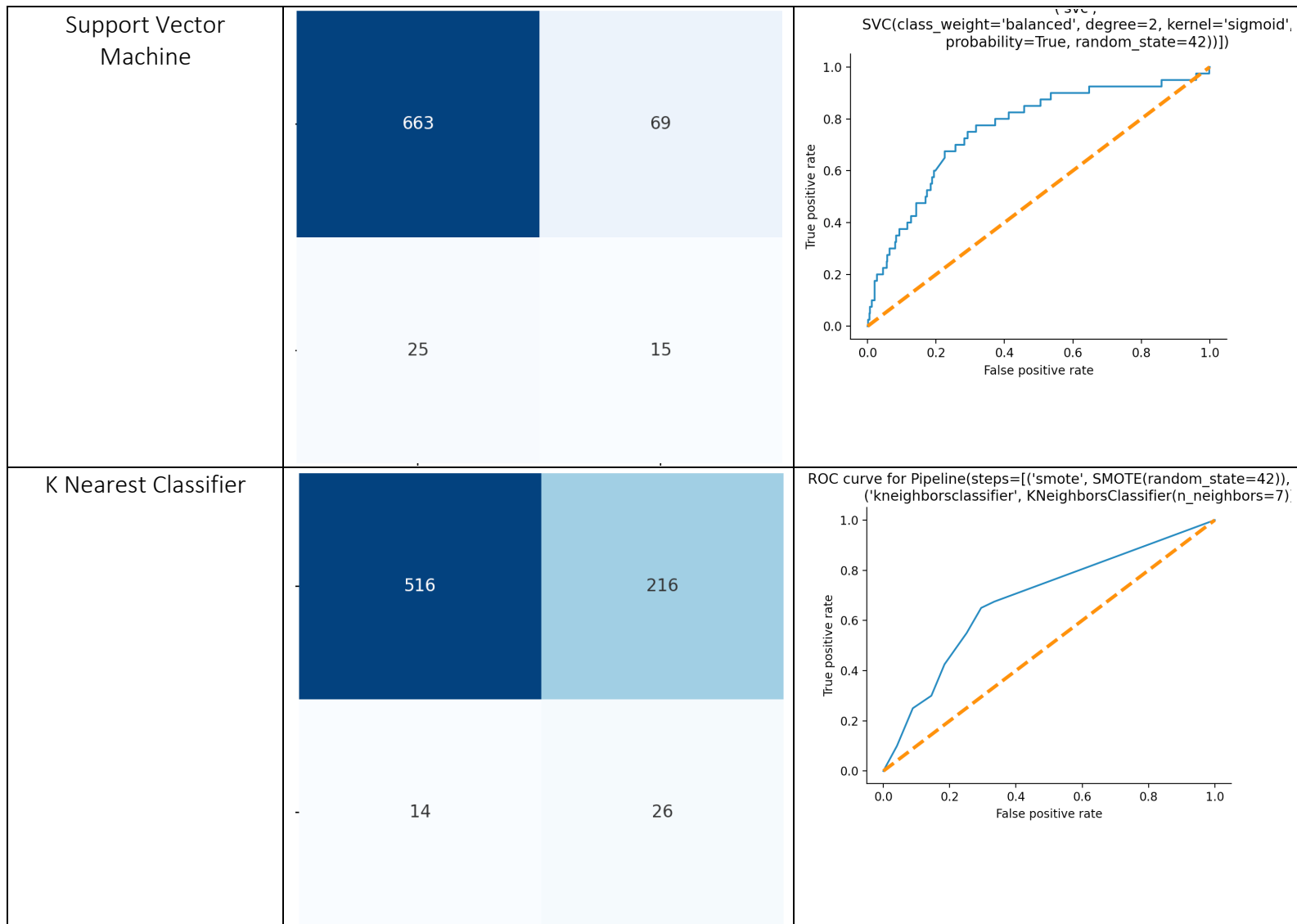
Table 7: Measurement Metrics for Validation set based on the best CV parameters after probability model prediction threshold tuning

	accuracy	precision	recall	Specificity	Sensitivity	f1	f2	roc_auc
Trivial Baseline	0.95	0.0	0.0	1.0	0.0	0.0	0.0	0.49435
Non-trivial Baseline	0.87	0.08696	0.15	0.91393	0.15	0.11009	0.131	0.53197
Logistic Regression	0.77	0.15423	0.775	0.76776	0.775	0.25726	0.42936	0.85451
Naive Bayes	0.74	0.13063	0.725	0.73634	0.725	0.22137	0.37958	0.80314
AdaBoost Classifier	0.64	0.11475	0.875	0.63115	0.875	0.20291	0.37634	0.79709
Support Vector Machine	0.88	0.17857	0.375	0.90574	0.375	0.24194	0.30738	0.76052
K Nearest Classifier	0.7	0.10744	0.65	0.70492	0.65	0.1844	0.32338	0.68248

Table 8: Confusion Matrix and ROC curve for Validation set based on best CV parameters after probability model prediction threshold tuning

	Confusion Matrix	ROC Curve									
Trivial Baseline	 <p>Confusion Matrix for Trivial Baseline:</p> <table border="1"> <thead> <tr> <th></th><th>Actual Negative</th><th>Actual Positive</th></tr> </thead> <tbody> <tr> <th>Predicted Negative</th><td>732</td><td>0</td></tr> <tr> <th>Predicted Positive</th><td>40</td><td>0</td></tr> </tbody> </table>		Actual Negative	Actual Positive	Predicted Negative	732	0	Predicted Positive	40	0	 <p>ROC curve for trivial</p> <p>The ROC curve for the trivial baseline is a diagonal line from (0,0) to (1,1), indicating no predictive power. The x-axis is False positive rate (0.0 to 1.0) and the y-axis is True positive rate (0.0 to 1.0).</p>
	Actual Negative	Actual Positive									
Predicted Negative	732	0									
Predicted Positive	40	0									
Non-trivial Baseline	 <p>Confusion Matrix for Non-trivial Baseline:</p> <table border="1"> <thead> <tr> <th></th><th>Actual Negative</th><th>Actual Positive</th></tr> </thead> <tbody> <tr> <th>Predicted Negative</th><td>669</td><td>63</td></tr> <tr> <th>Predicted Positive</th><td>34</td><td>6</td></tr> </tbody> </table>		Actual Negative	Actual Positive	Predicted Negative	669	63	Predicted Positive	34	6	 <p>ROC curve for KNeighborsClassifier(n_neighbors=1)</p> <p>The ROC curve for the KNeighborsClassifier(n_neighbors=1) model shows a blue line above the orange dashed diagonal line, indicating better predictive performance than the trivial baseline. The x-axis is False positive rate (0.0 to 1.0) and the y-axis is True positive rate (0.0 to 1.0).</p>
	Actual Negative	Actual Positive									
Predicted Negative	669	63									
Predicted Positive	34	6									





We could see from the plots above: The Logistic Regression has a very good ROC curve; then is Naïve Bayes, SVM and AdaBoost; KNN performs worst to compare with other models, but all of them perform better than the baseline. Logistic Regression has good f2 score than others, and both Specificity and Sensitivity are good. But SVM only has good Specificity but recall and sensitivity are low; AdaBoost only has good Sensitivity and has a low accuracy; Naïve Bayes overall performs better than others except Logistic Regression. From the Confusion Metrics, we found that AdaBoost has a high recall, but the non-stroke class has a lower recall than other models. From Recall of both of the classes, we could see Naïve Bayes is not better than Logistic Regression.

3. Summary and conclusions

Based on the table of measurement metrics, we could find that Logistic Regression has the best overall performance to compare with the baselines and other models. All of the models performed better than the two baseline models, except for the accuracy

score. Accuracy score in the stroke prediction is not reliable. Recall, Sensitivity, Specificity, fbeta ($\beta > 1$) and roc auc are more reliable in this case.