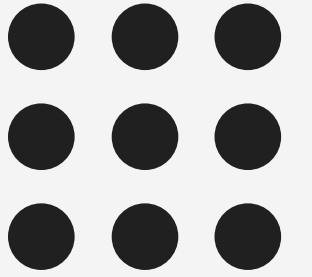


ANDREEA STROIA  
HALA ALBAHLOUL  
HRITIKA KATHURIA

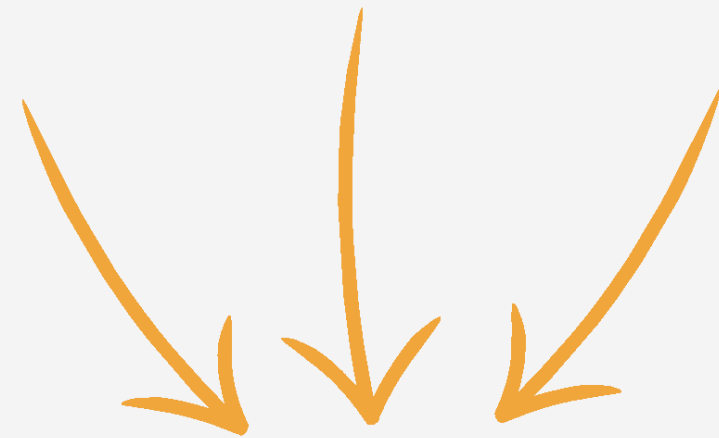
# SUPERVISED CLASSIFICATION METHODS



# Overview



3 datasets

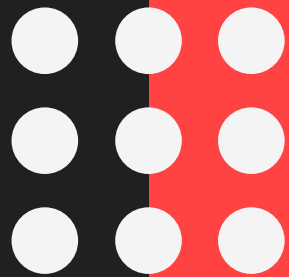


*Diabetes prediction*

*Heart attack prediction*

*MNIST*

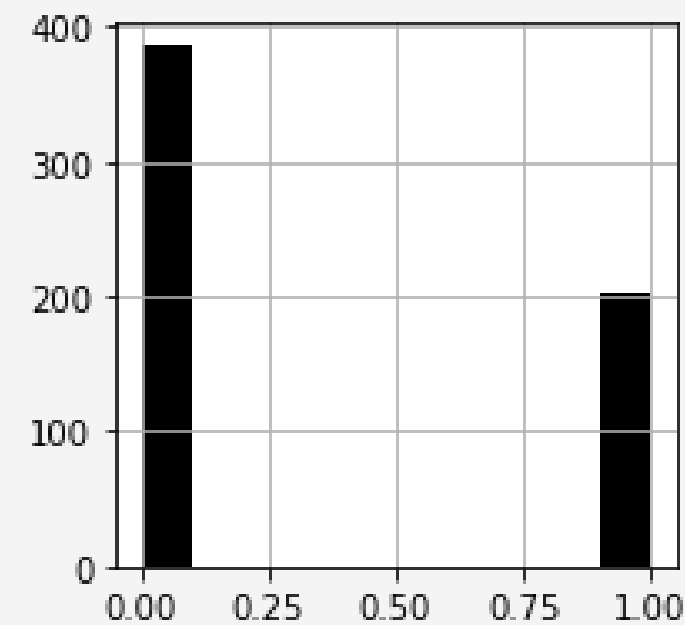
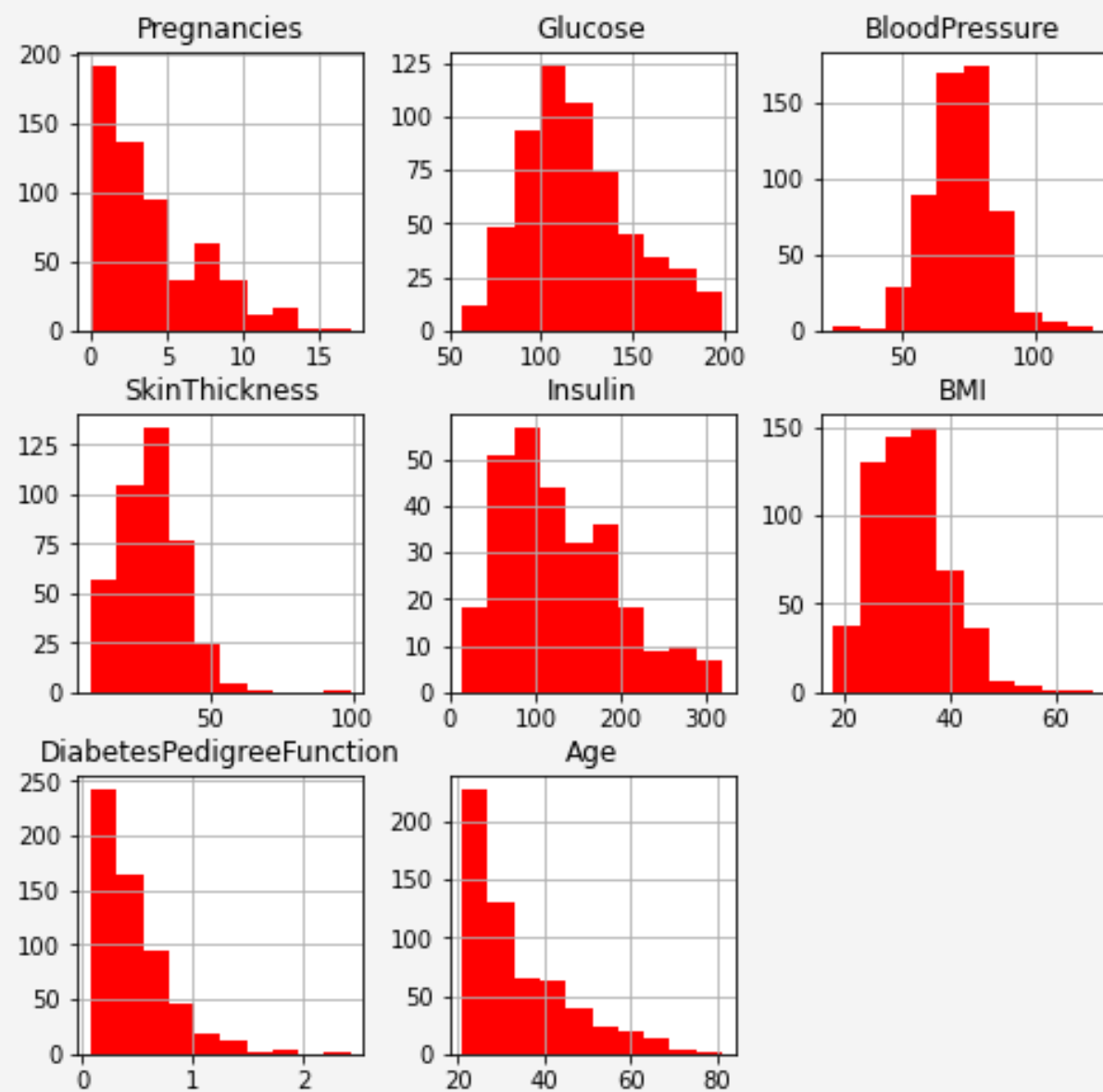
- EDA
- Classical method of classification
- SVM
- Random forest



# Diabetes prediction

768 observations  
9 variables

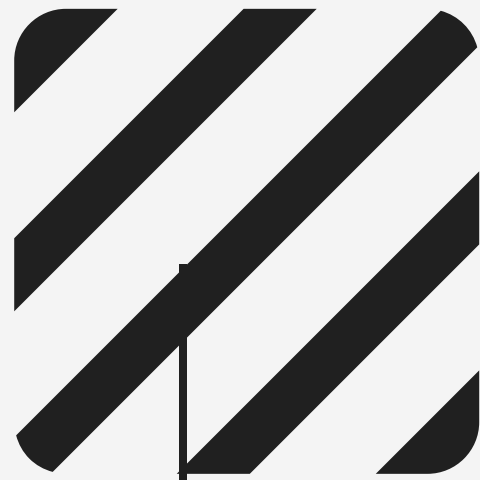
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



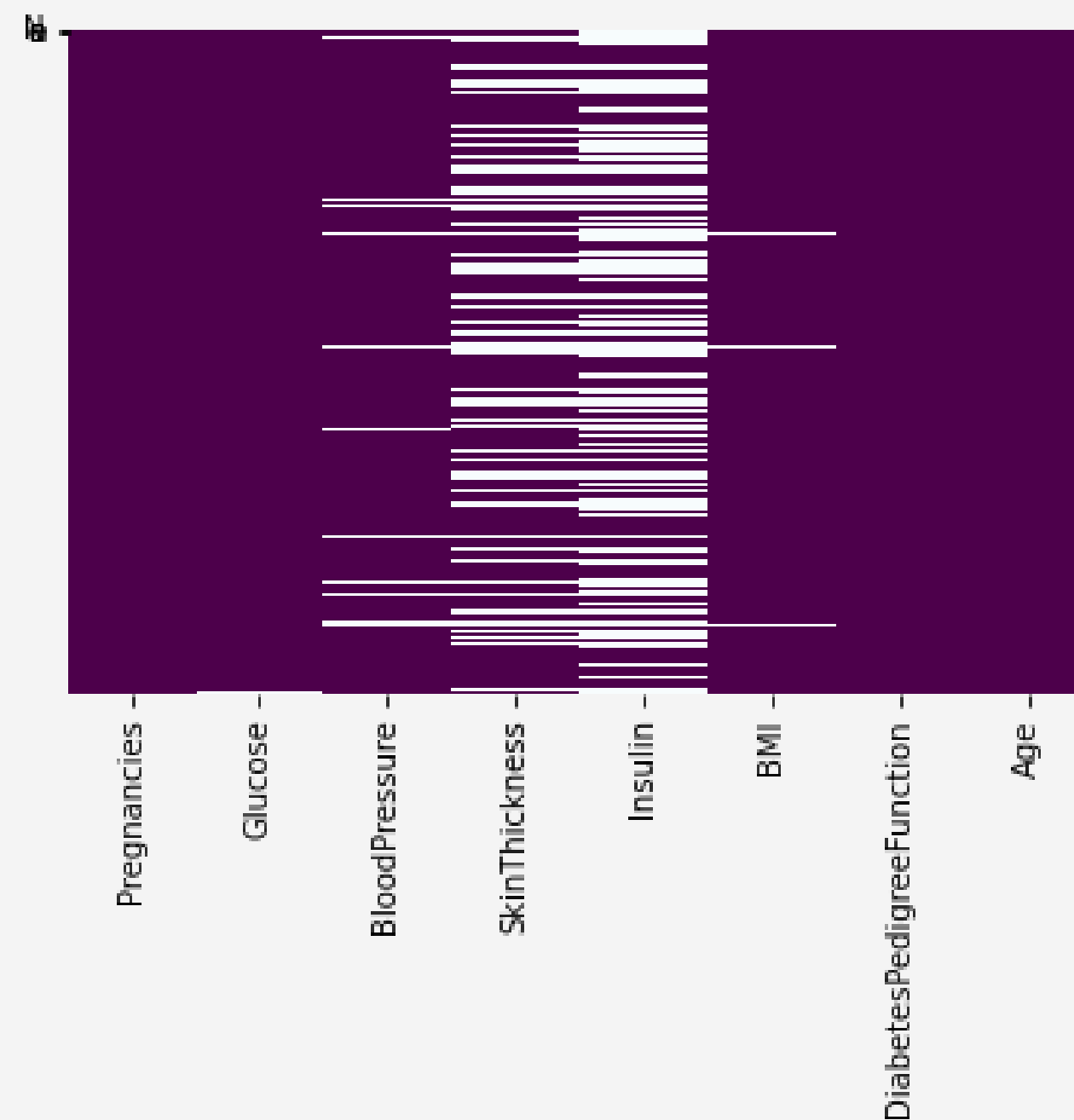
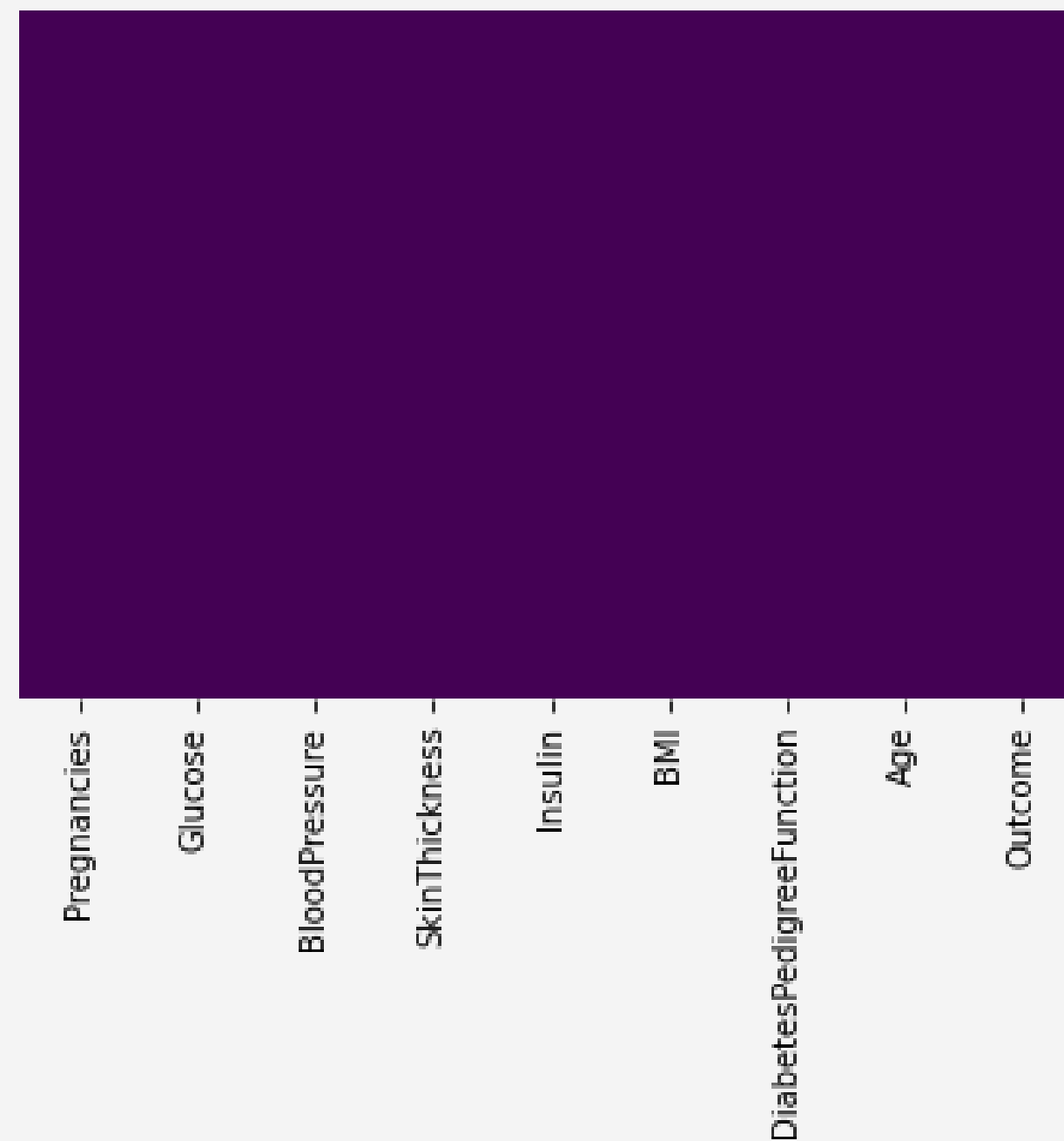
# EDA

- Removing outliers
- Filling the missing values

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
<b>count</b>	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
<b>mean</b>	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
<b>std</b>	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
<b>min</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
<b>25%</b>	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
<b>50%</b>	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
<b>75%</b>	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
<b>max</b>	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000



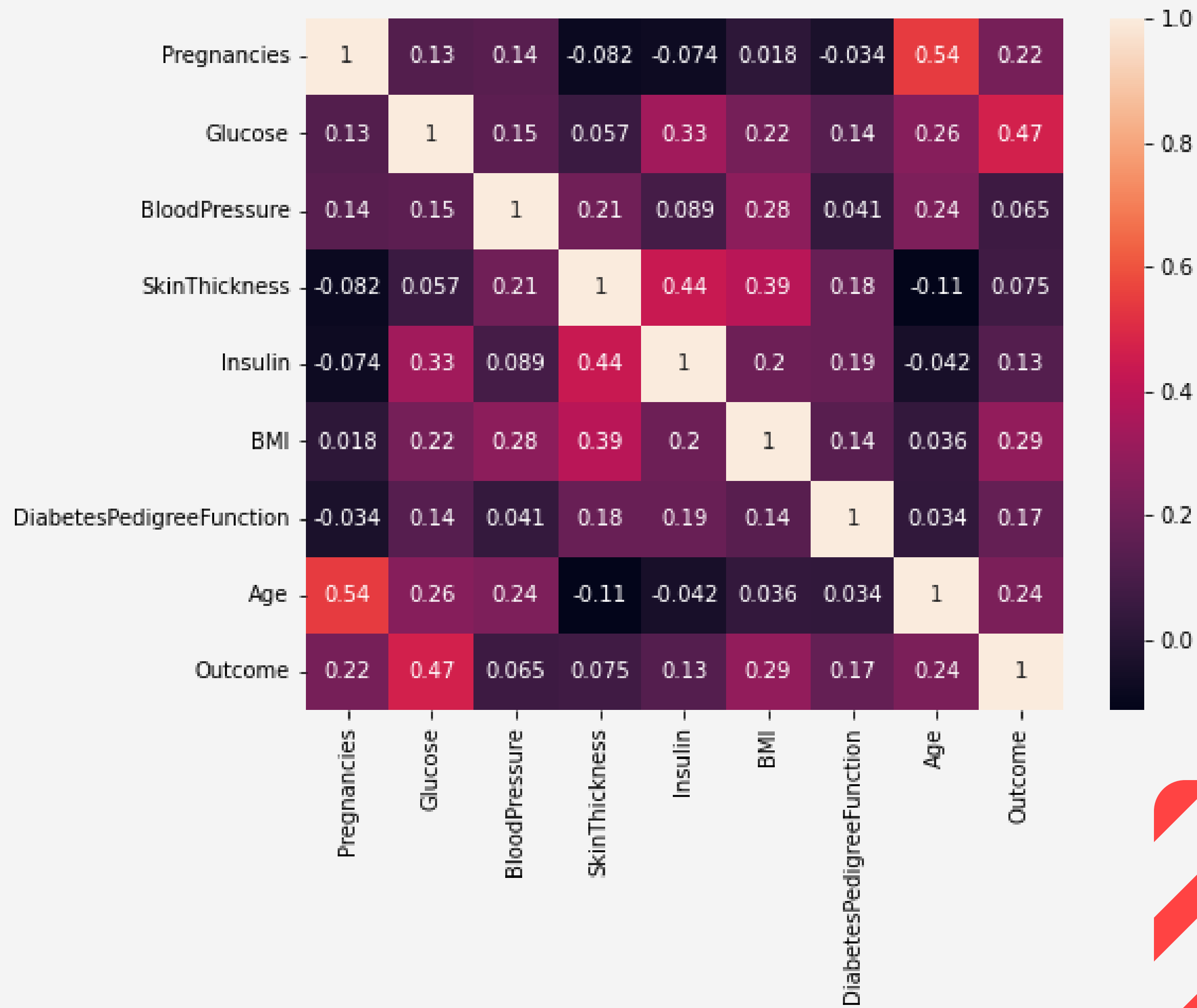
Missing  
values



Standardization

```
[ -0.53498699, -0.27205017, -0.53415007, ..., -1.08499402,  
  -0.48781118, -0.77557037],  
[ -1.12751031, -0.40593694, -0.36341983, ..., -0.75115541,  
   0.97088533, -0.09493526],  
[  1.53884464,  0.66515725,  1.85607334, ...,  0.03263958,  
   0.80982093,  1.01109679],
```

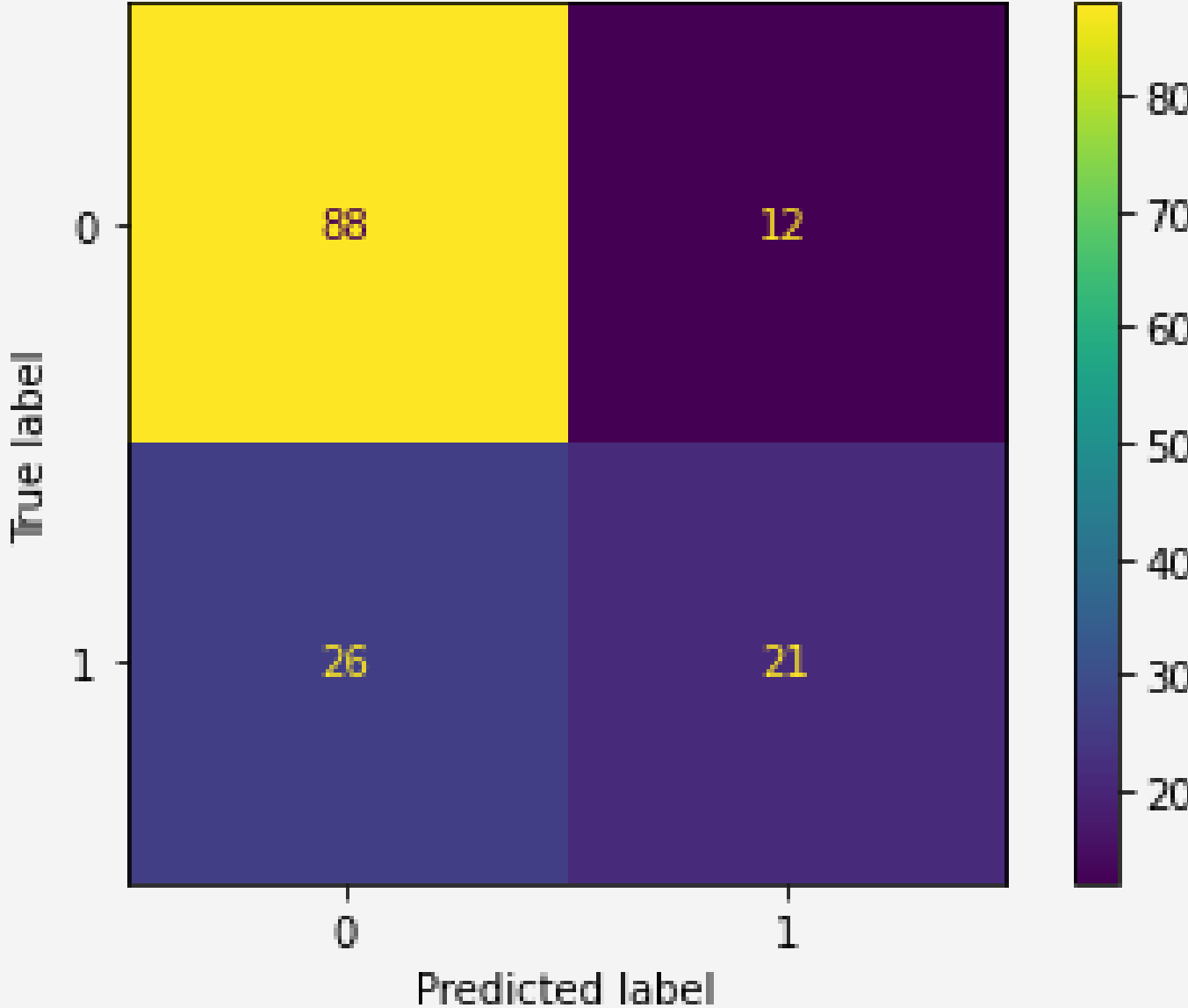
# CORRELATION



# Logistic Regression

Predict if a person has diabetes or not

	precision	recall	f1-score	support
0	0.77	0.88	0.82	100
1	0.64	0.45	0.52	47
accuracy			0.74	147
macro avg	0.70	0.66	0.67	147
weighted avg	0.73	0.74	0.73	147



# SVM

linear



0.787

poly



0.798

rbf



0.827

sigmoid



0.674

Accuracy  
scores for  
train

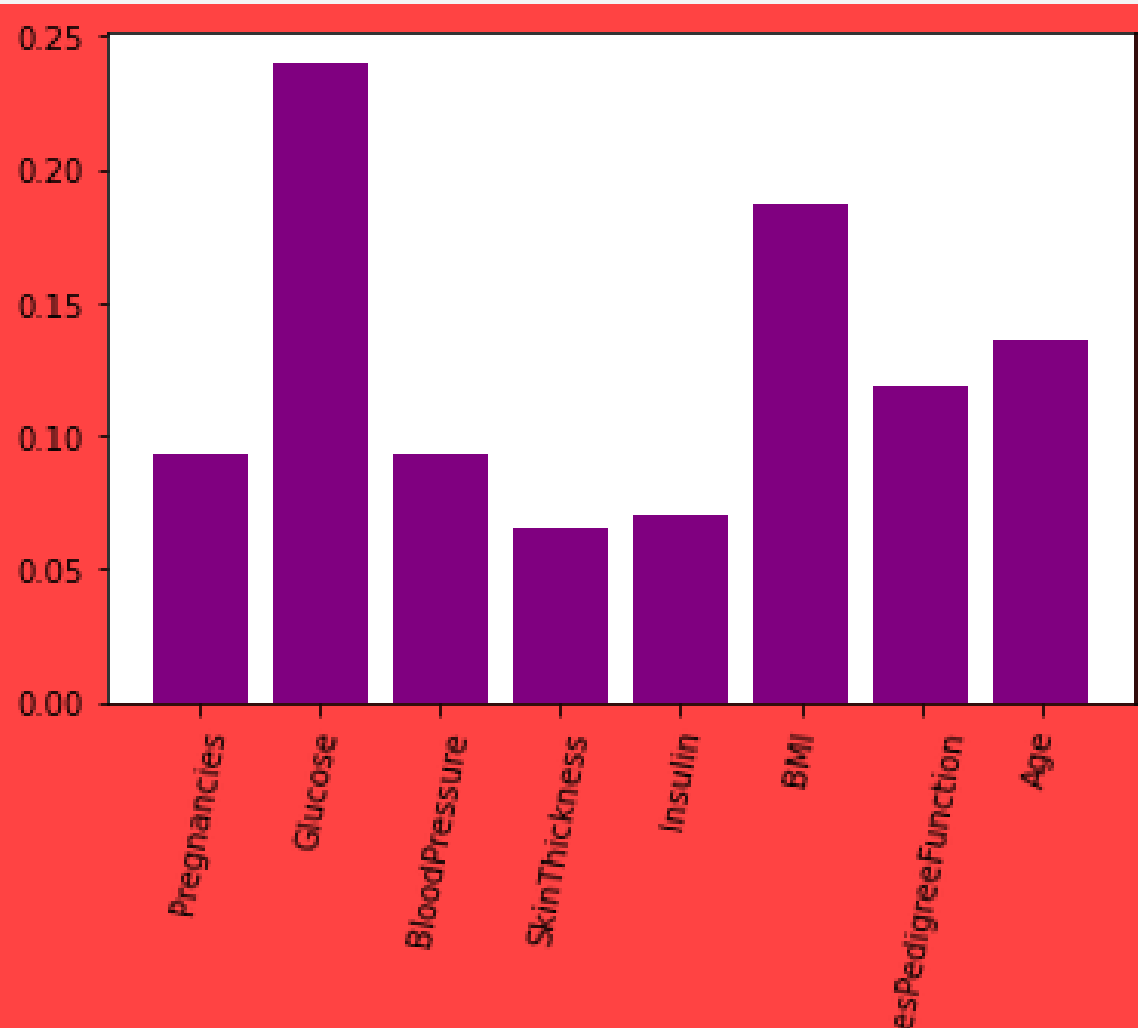
	precision	recall	f1-score	support
0	0.72	0.93	0.81	100
1	0.61	0.23	0.34	47
accuracy			0.71	147
macro avg	0.67	0.58	0.58	147
weighted avg	0.69	0.71	0.66	147



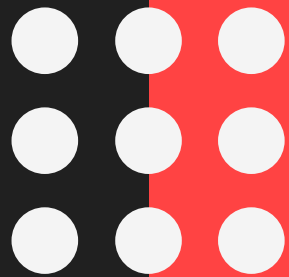
# Random forest

	precision	recall	f1-score	support
0	0.72	0.92	0.81	135
1	0.77	0.43	0.55	86
accuracy			0.73	221
macro avg	0.74	0.67	0.68	221
weighted avg	0.74	0.73	0.71	221

# Feature importance



Feature	Importance
Pregnancies	0.09
Glucose	0.23
BloodPressure	0.09
SkinThickness	0.06
Insulin	0.06
BMI	0.18
DiabetesPedigreeFunction	0.11
Age	0.13

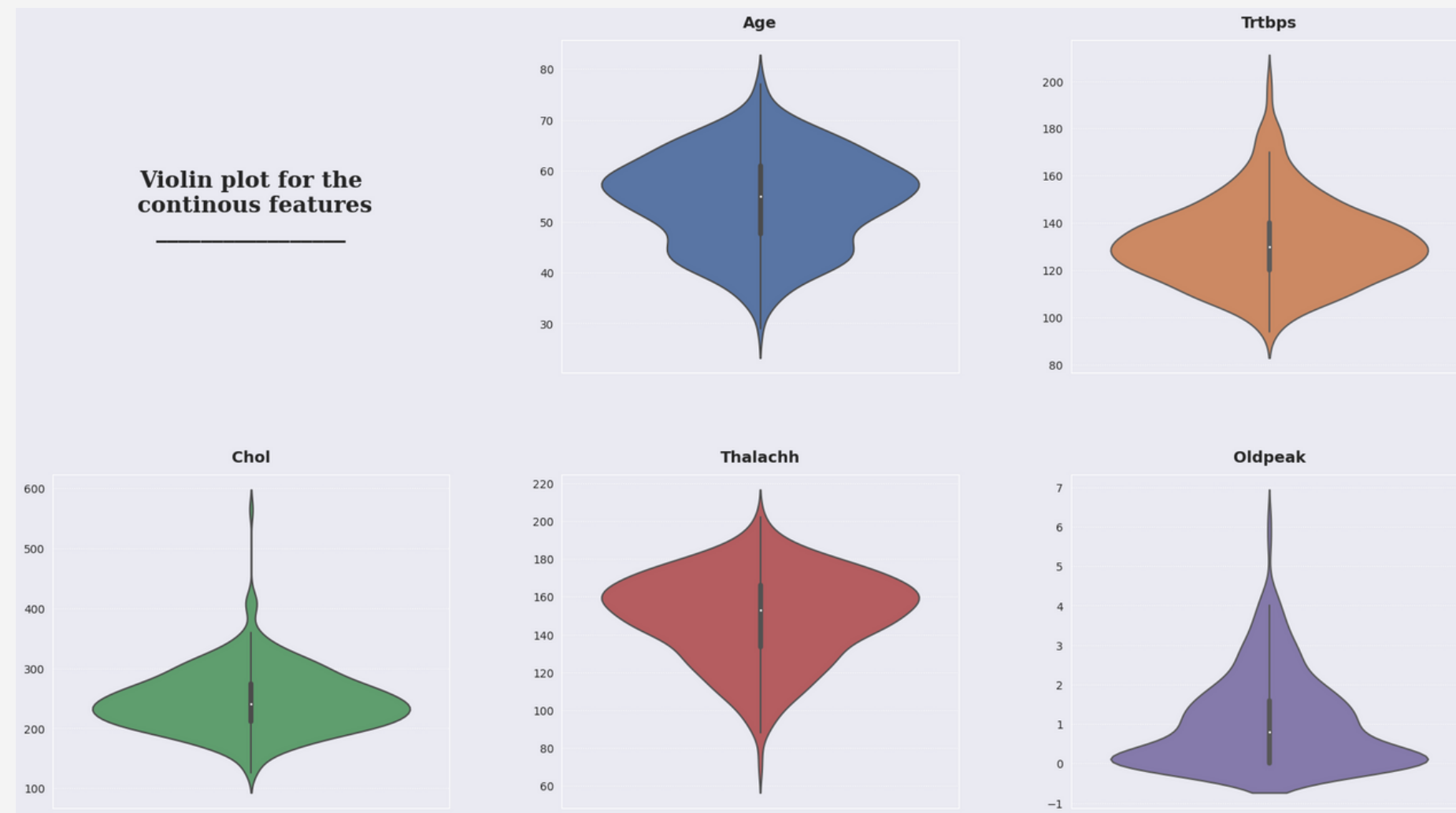


# Heart attack prediction

303 observations  
14 variables

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

**Violin plot for the  
continous features**



``cp`` - Chest pain type

``trtbps`` - Resting blood pressure (in mm Hg)

``chol`` - Cholestoral in mg/dl

``fbs`` - (fasting blood sugar > 120 mg/dl)

``restecg`` - Resting electrocardiographic

``thalachh`` - Maximum heart rate achieved

``oldpeak`` - Previous peak

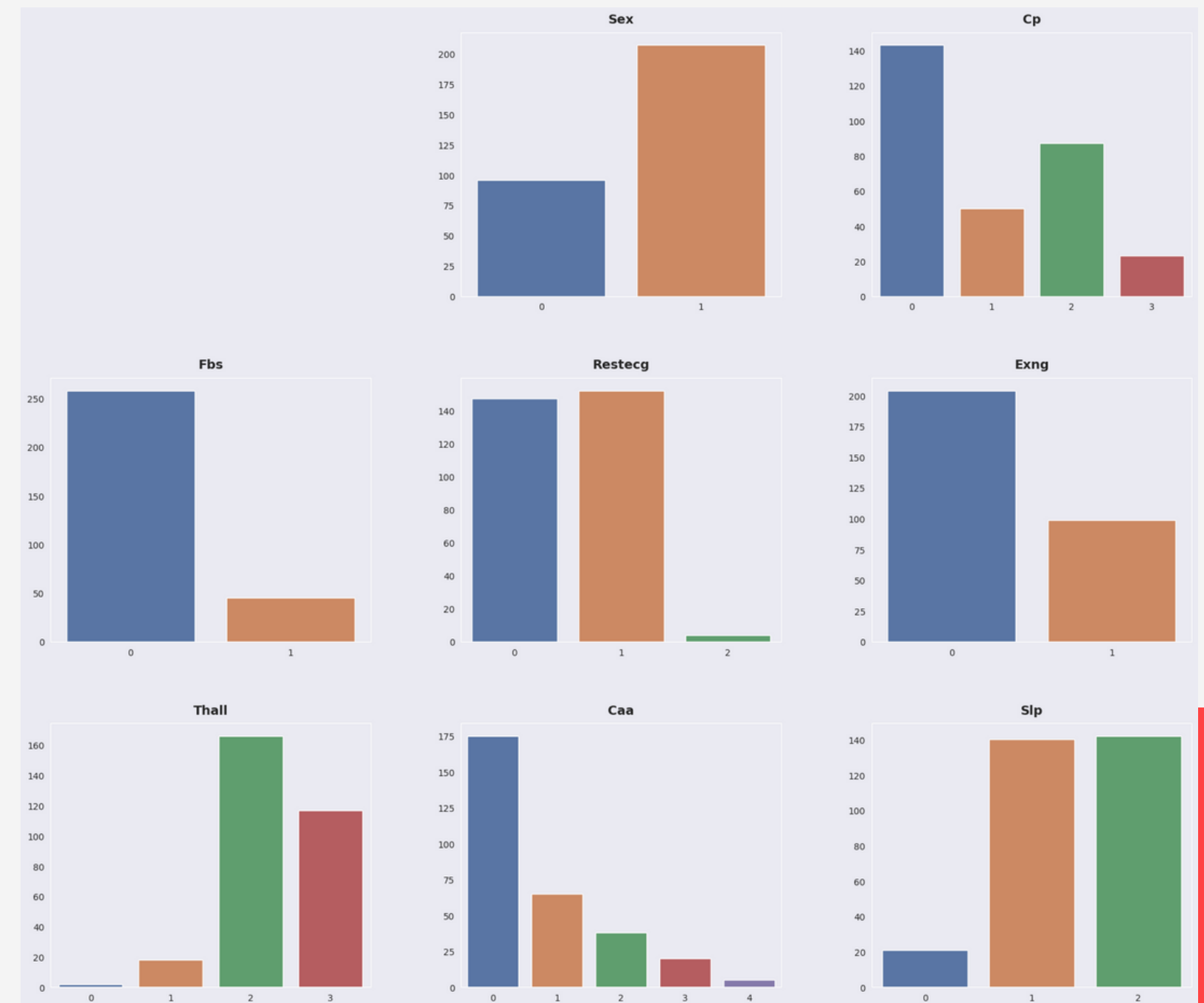
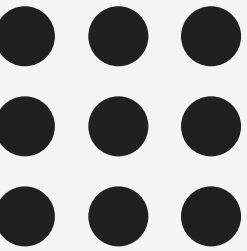
``slp`` - Slope

``caa`` - Number of major vessels

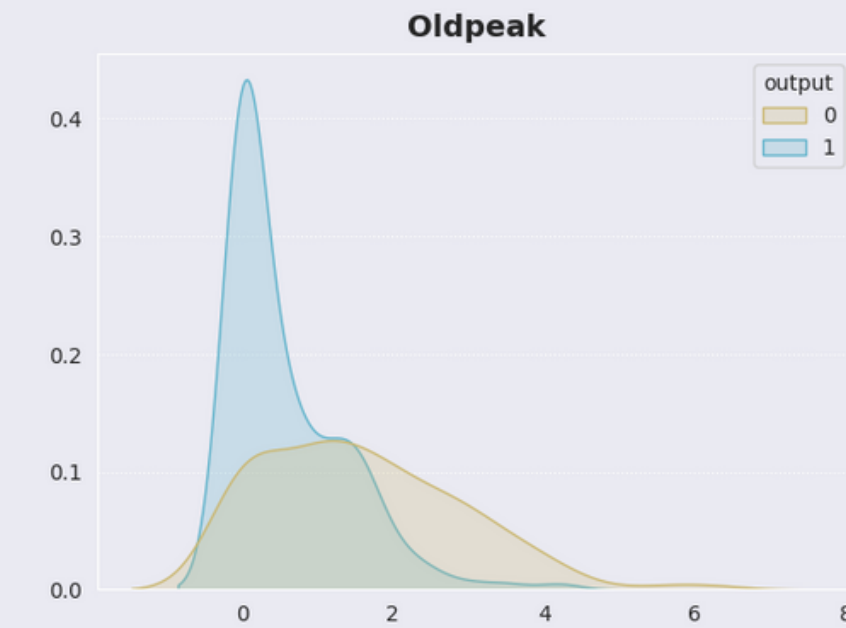
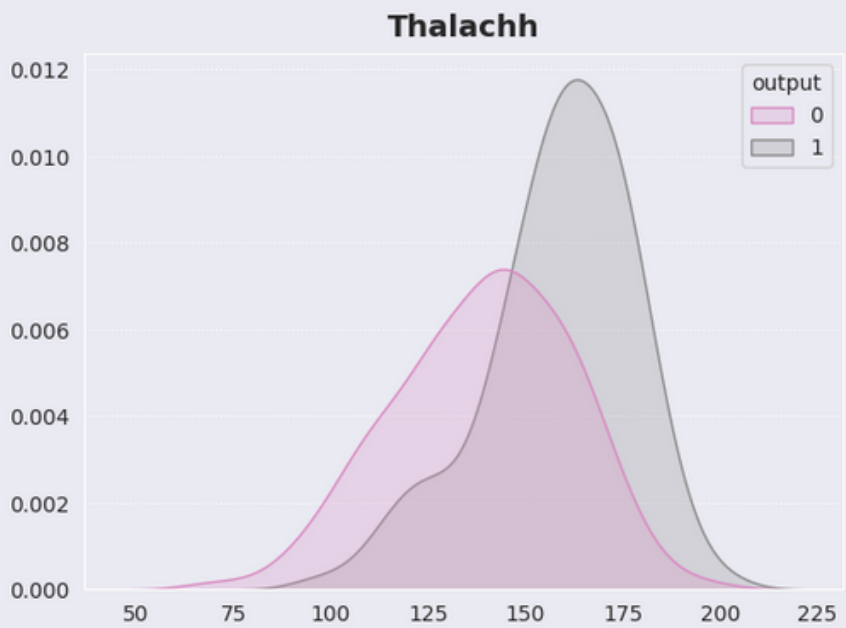
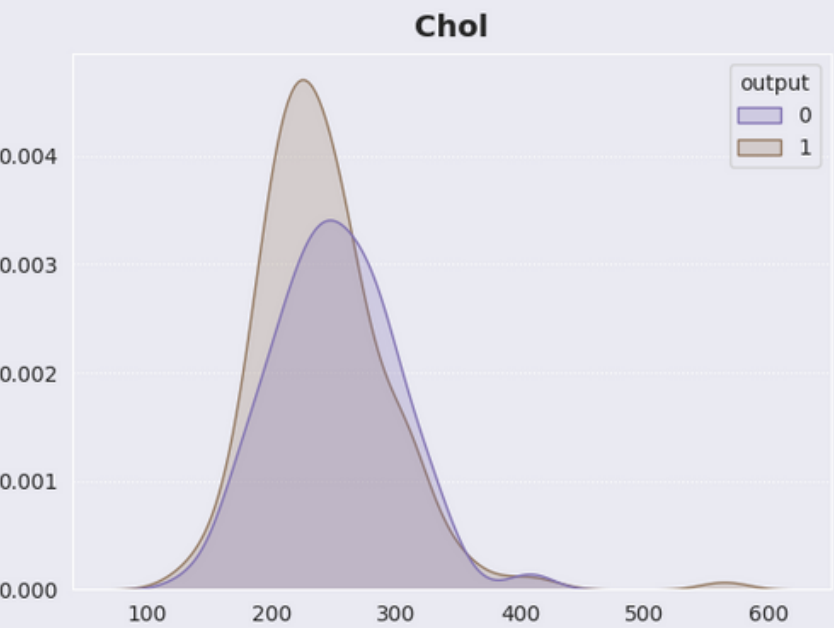
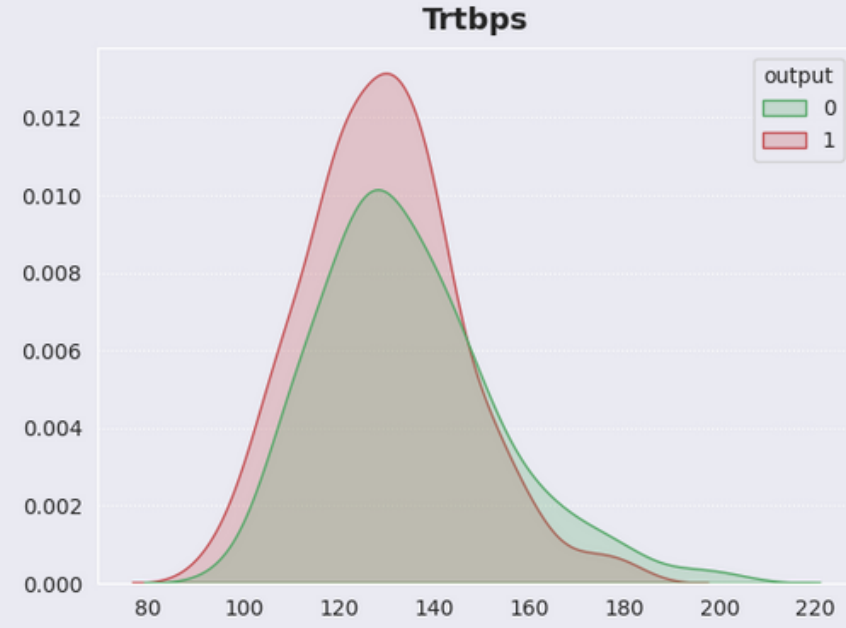
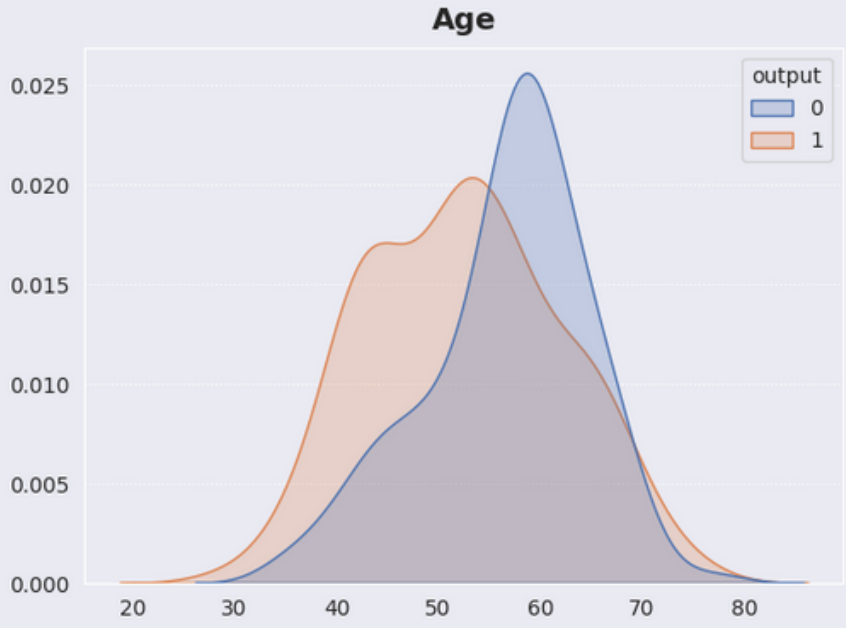
``thall`` - Thallium Stress Test

``exng`` - Exercise induced angina

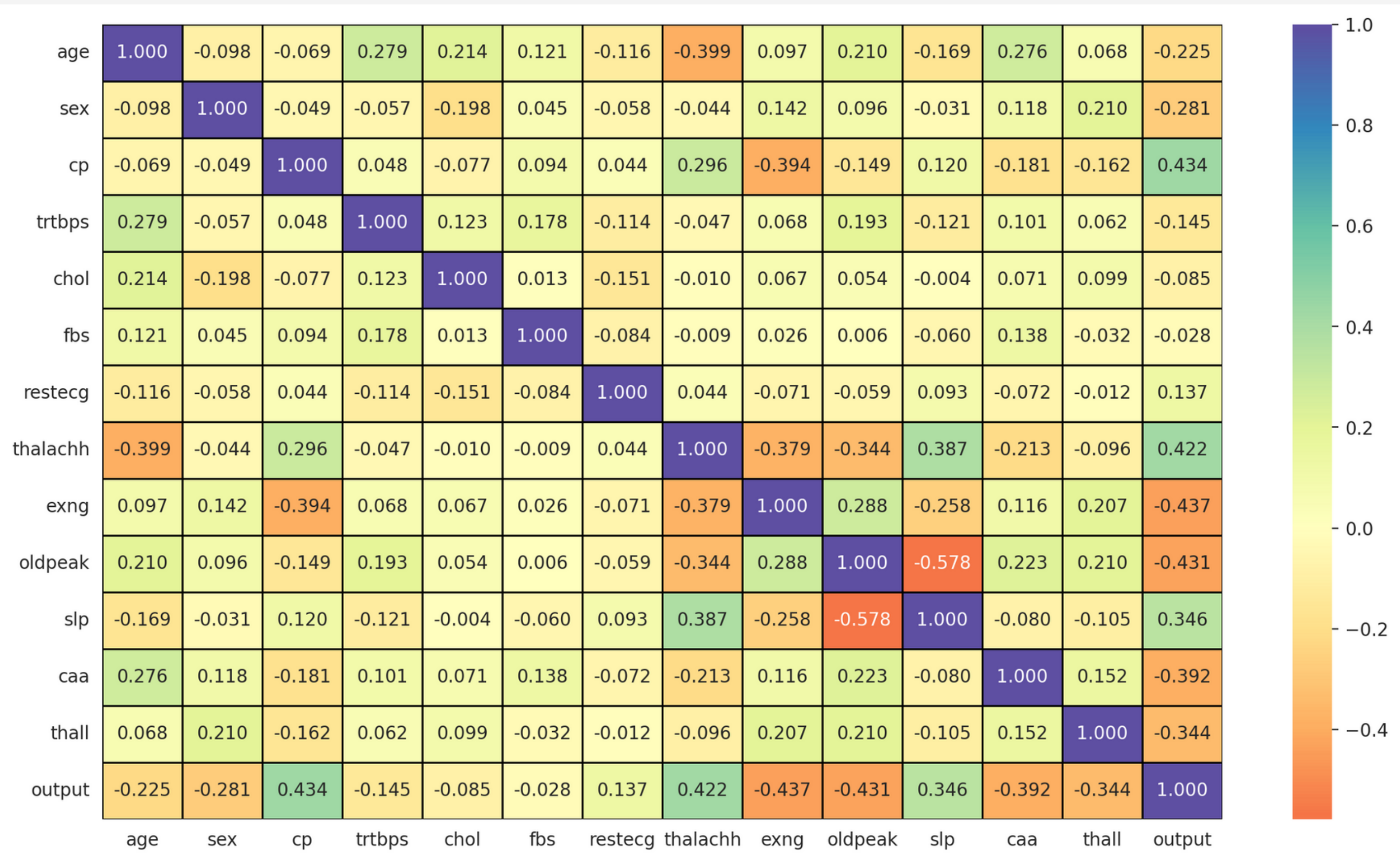
# EDA



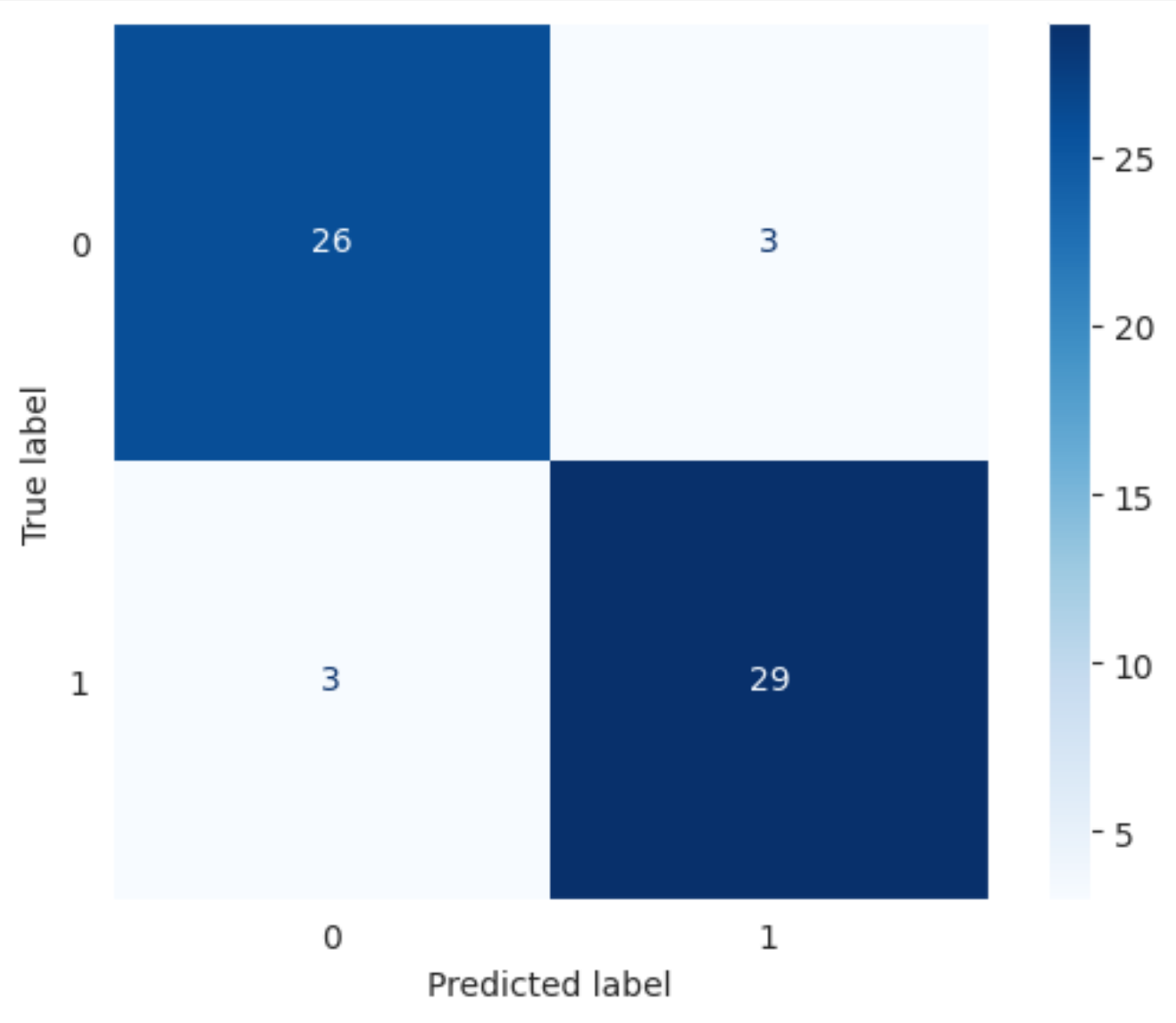
Distribution of continous variables by output



- Trtbps and Chol are not likely to have correlation.
- Age and Thalachh might have weak correlation.
- Oldpeak is likely to be correlated.



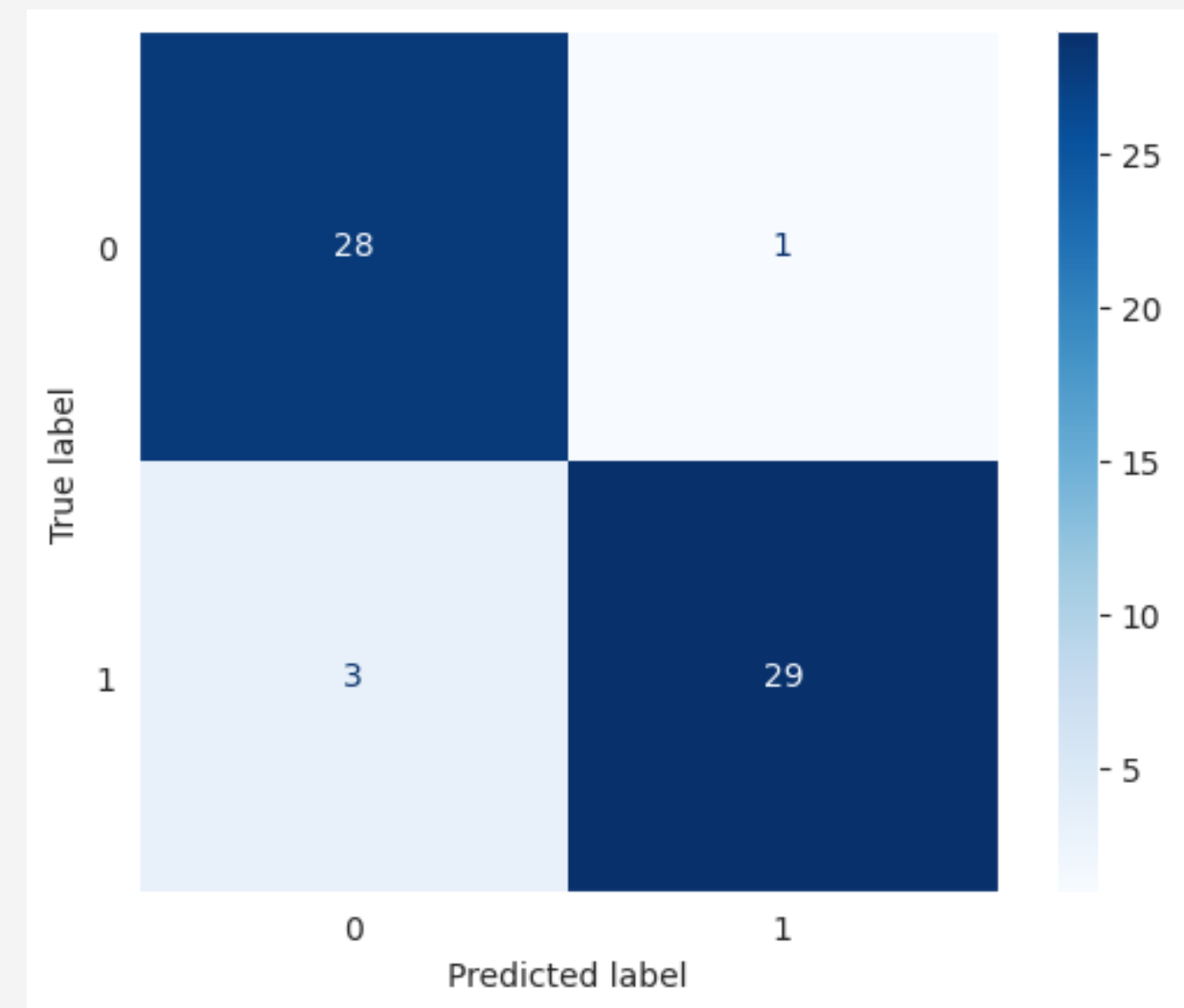
# SVM



	precision	recall	f1-score	support
0	0.90	0.90	0.90	29
1	0.91	0.91	0.91	32
accuracy			0.90	61
macro avg	0.90	0.90	0.90	61
weighted avg	0.90	0.90	0.90	61

finding the optimal parameter values  
by GridSearchCV

The best params are :  
'C'(Regularization): 3  
'gamma': 0.1



The test accuracy score of SVM after hyper-parameter tuning is 0.9344262295081968

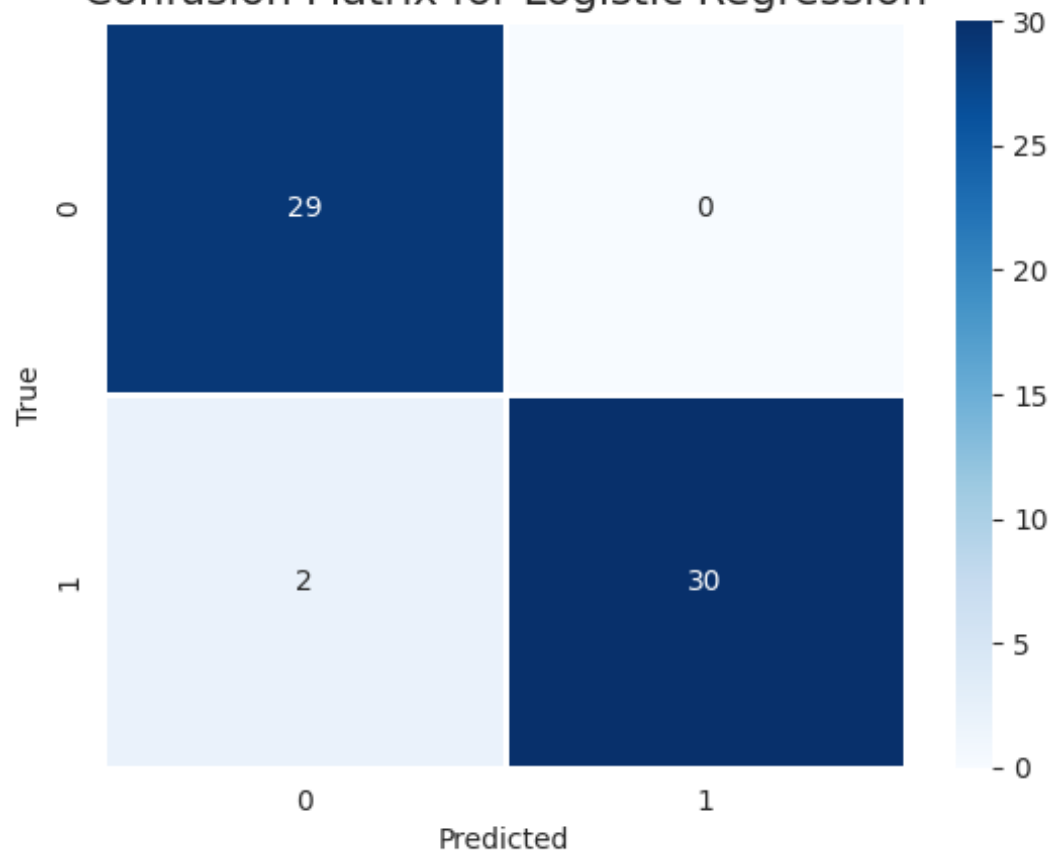
	precision	recall	f1-score	support
0	0.90	0.97	0.93	29
1	0.97	0.91	0.94	32
accuracy			0.93	61
macro avg	0.93	0.94	0.93	61
weighted avg	0.94	0.93	0.93	61

## Logistic Regression

-----

Accuracy: 0.96721  
Precision: 1.0  
Recall: 0.9375  
F1: 0.96774

Confusion Matrix for Logistic Regression

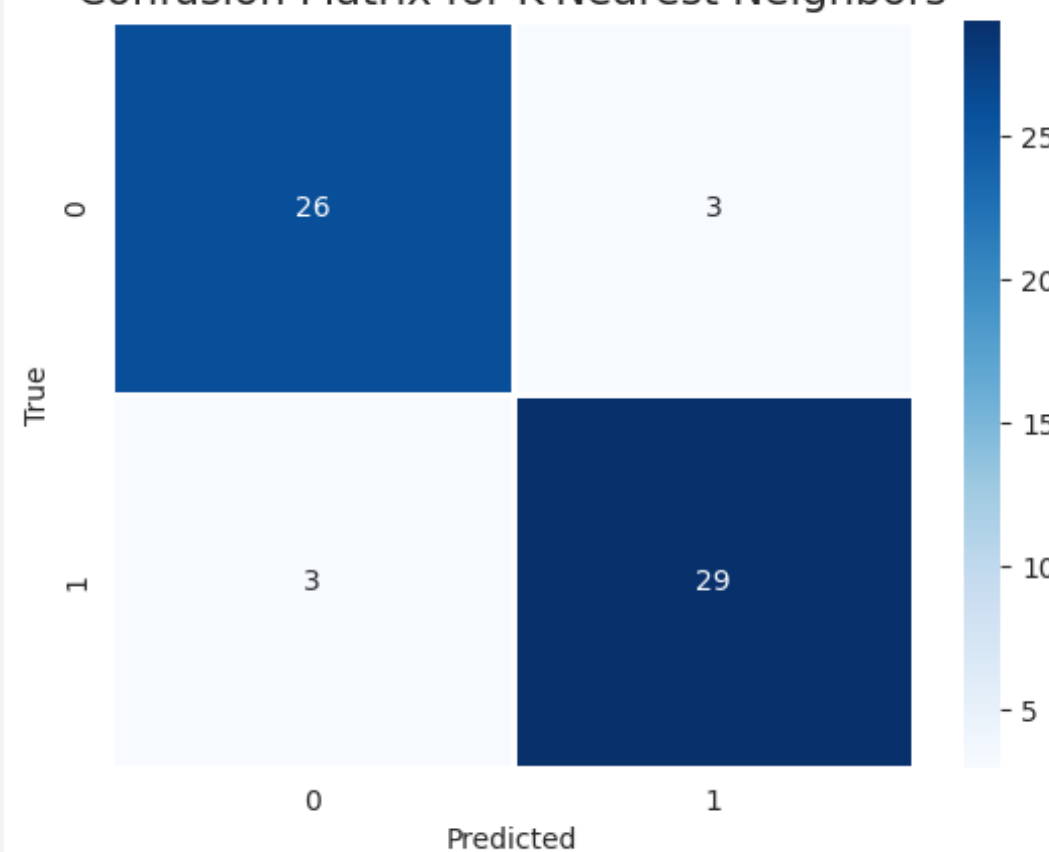


## K-Nearest Neighbors

-----

Accuracy: 0.90164  
Precision: 0.90625  
Recall: 0.90625  
F1: 0.90625

Confusion Matrix for K-Nearest Neighbors

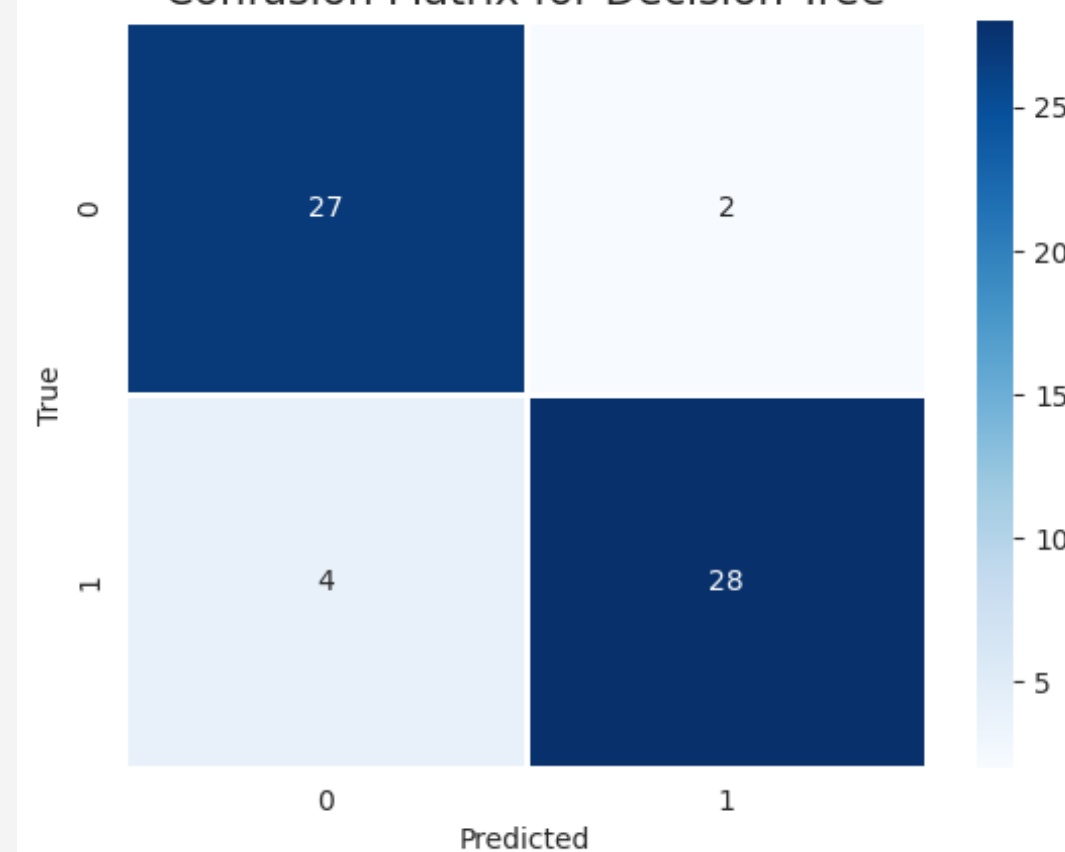


## Decision Tree

-----

Accuracy: 0.90164  
Precision: 0.93333  
Recall: 0.875  
F1: 0.90323

Confusion Matrix for Decision Tree





## Random Forest

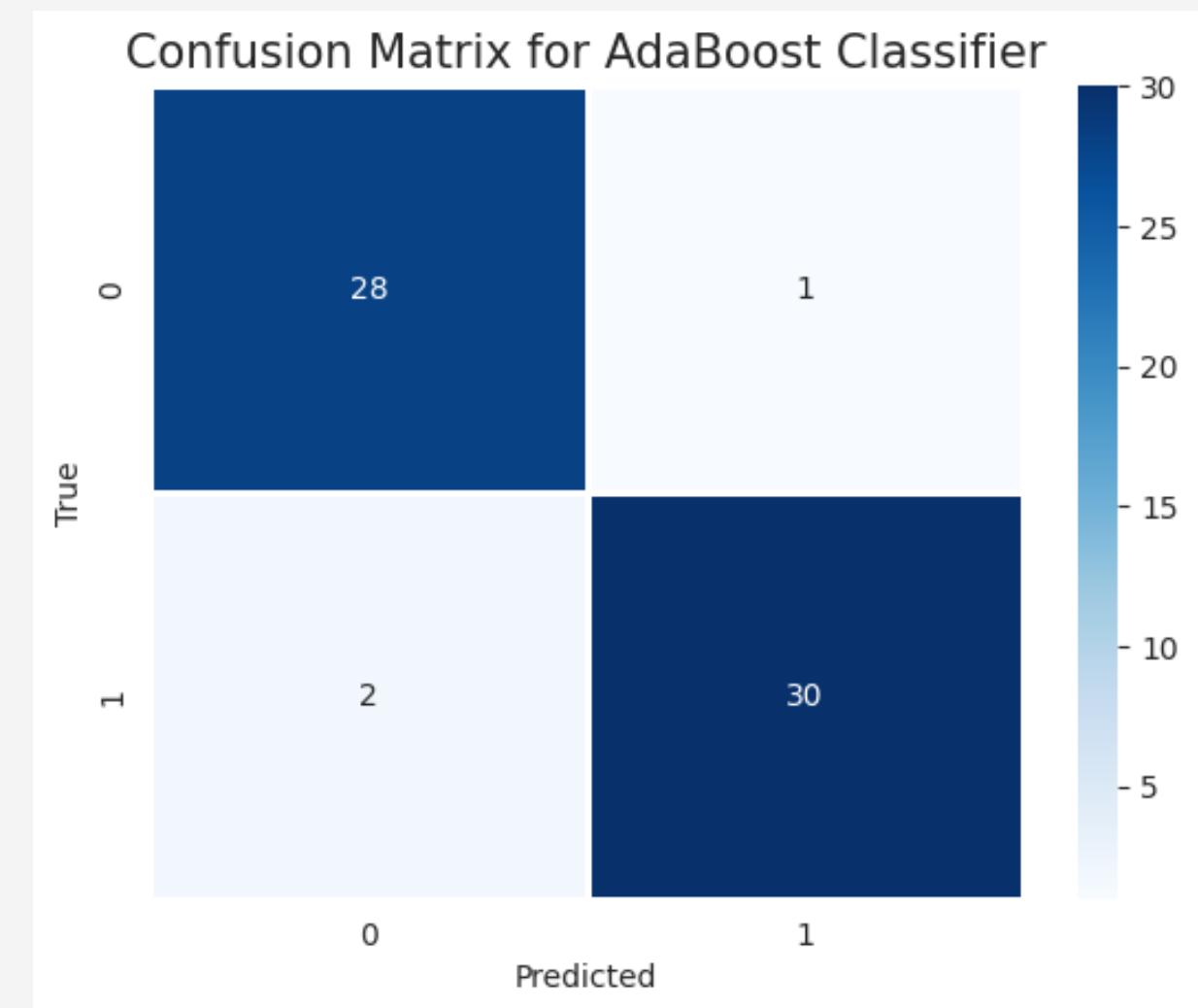
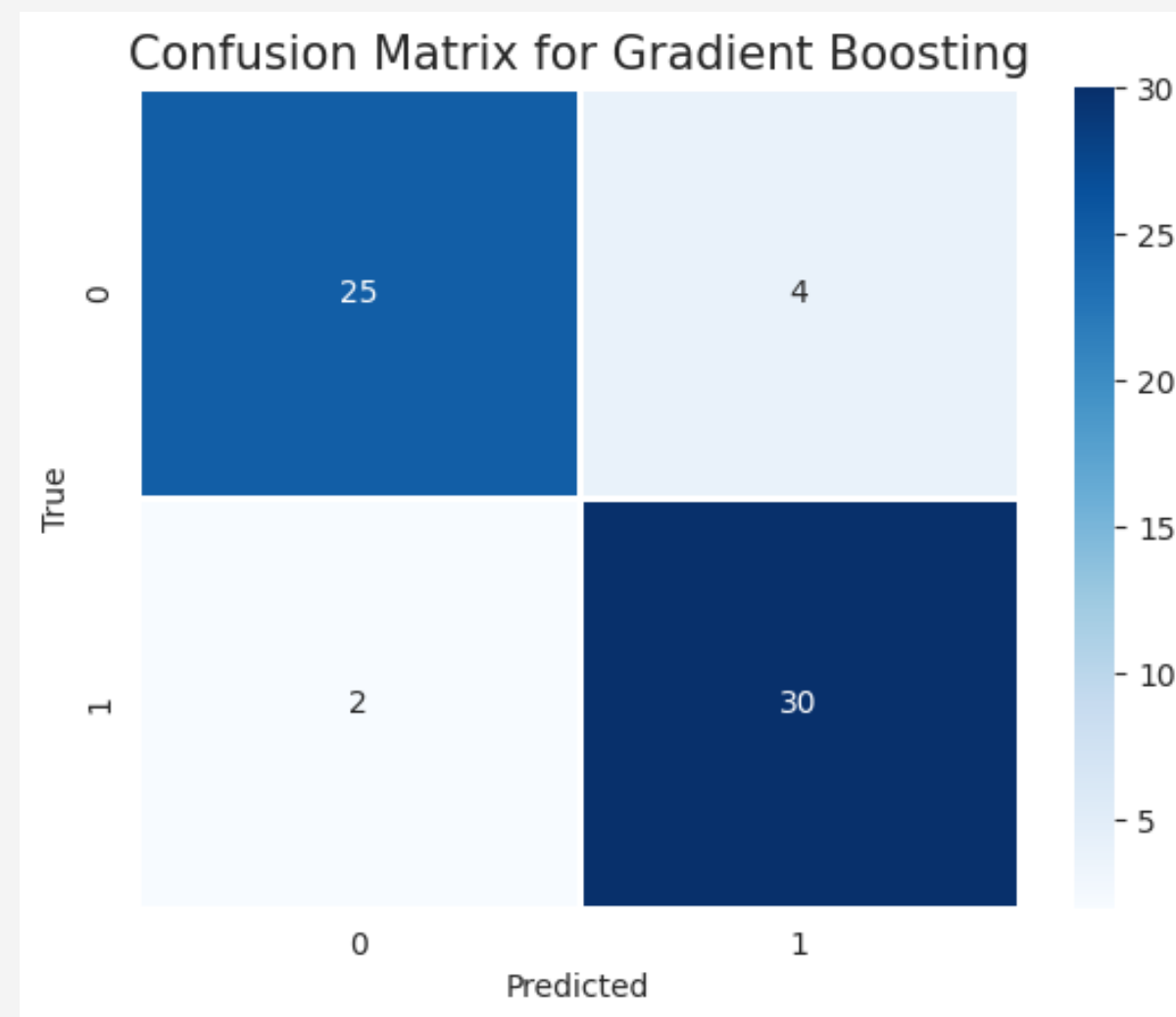
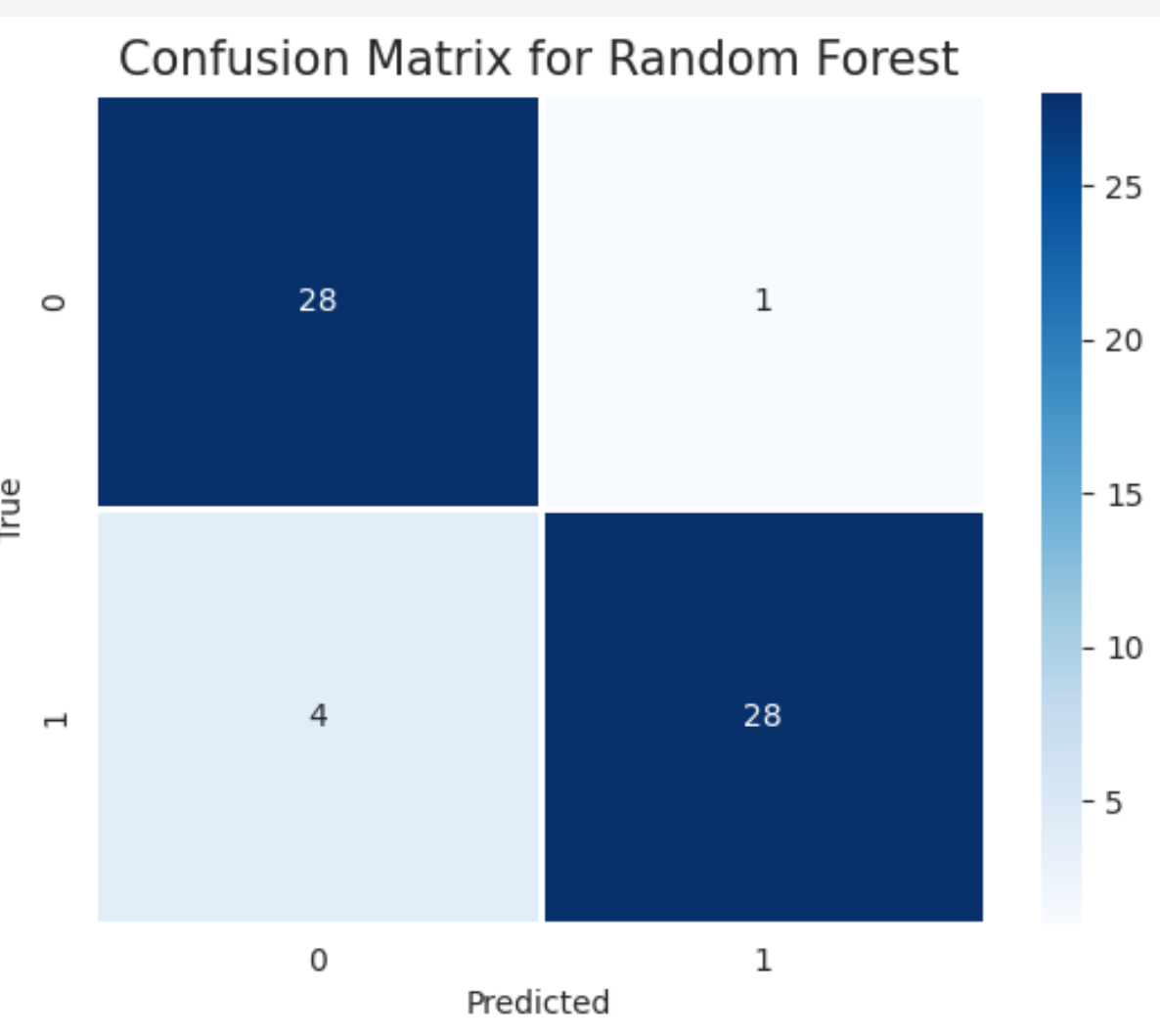
Accuracy: 0.91803  
Precision: 0.96552  
Recall: 0.875  
F1: 0.91803

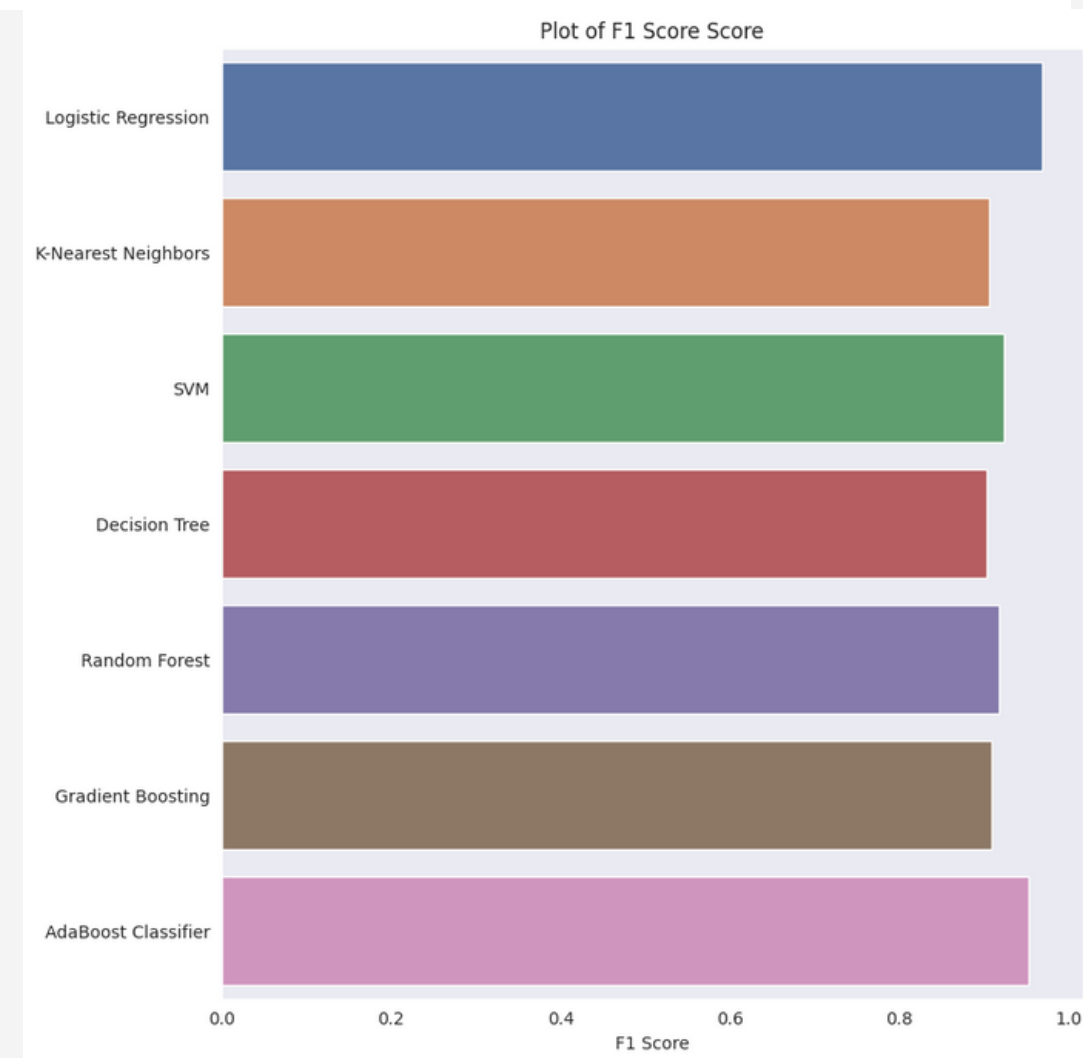
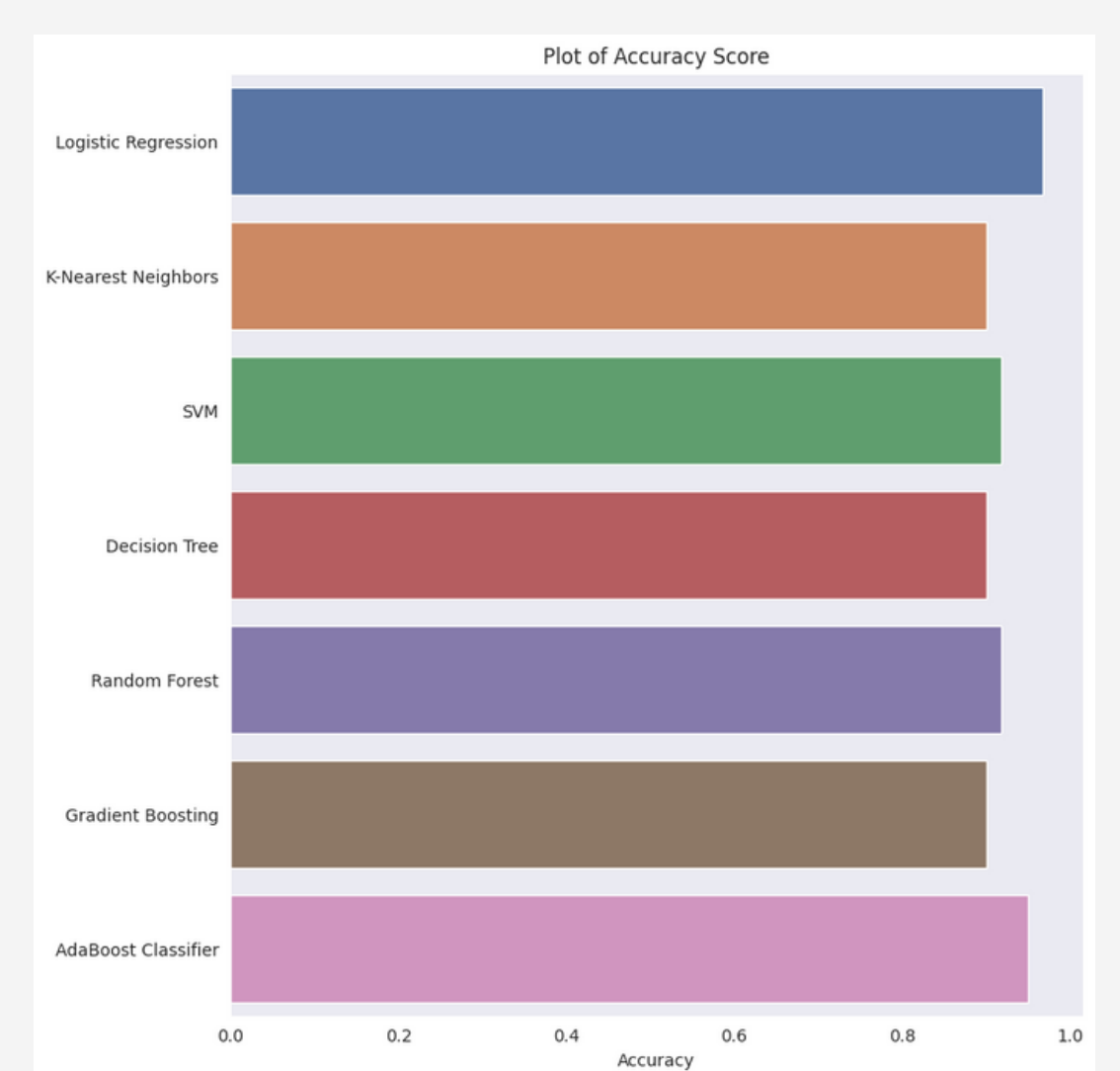
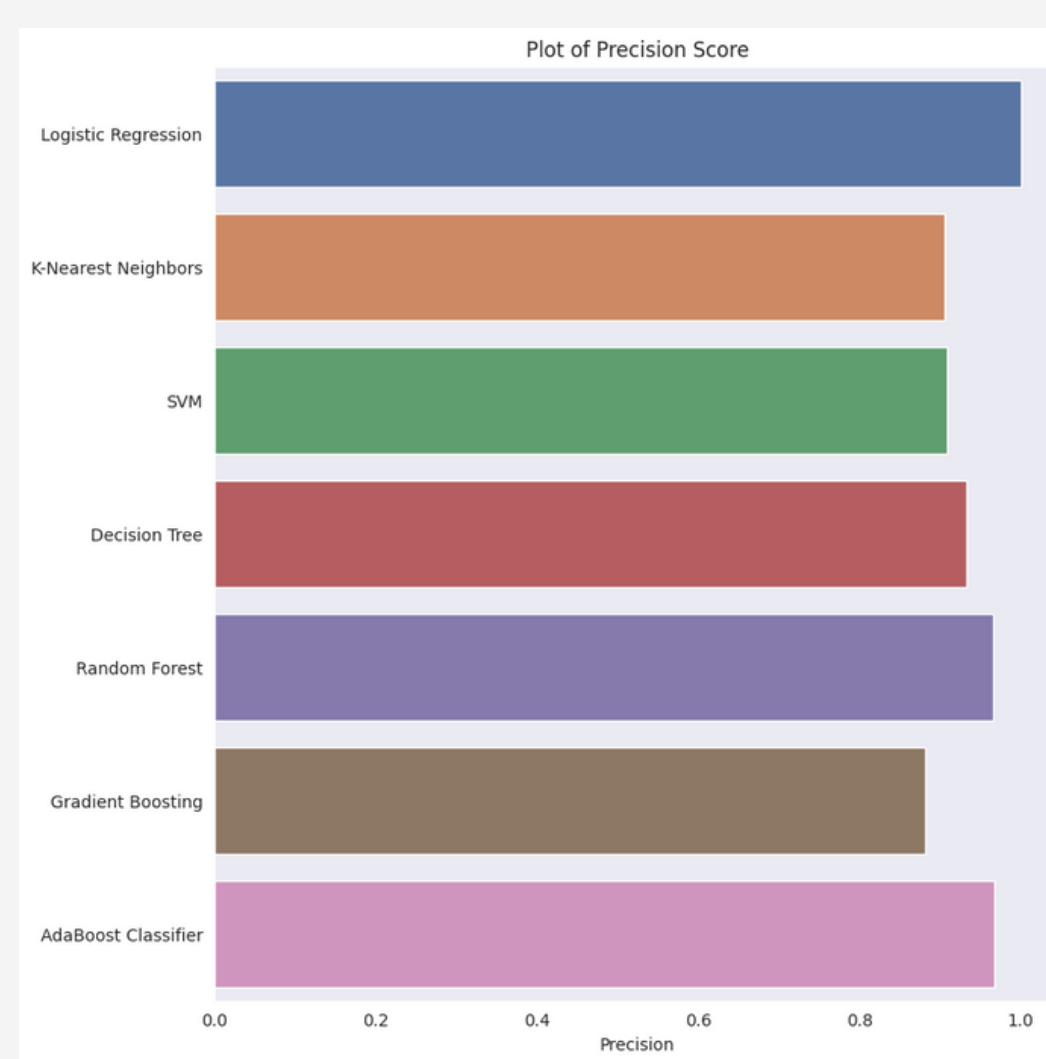
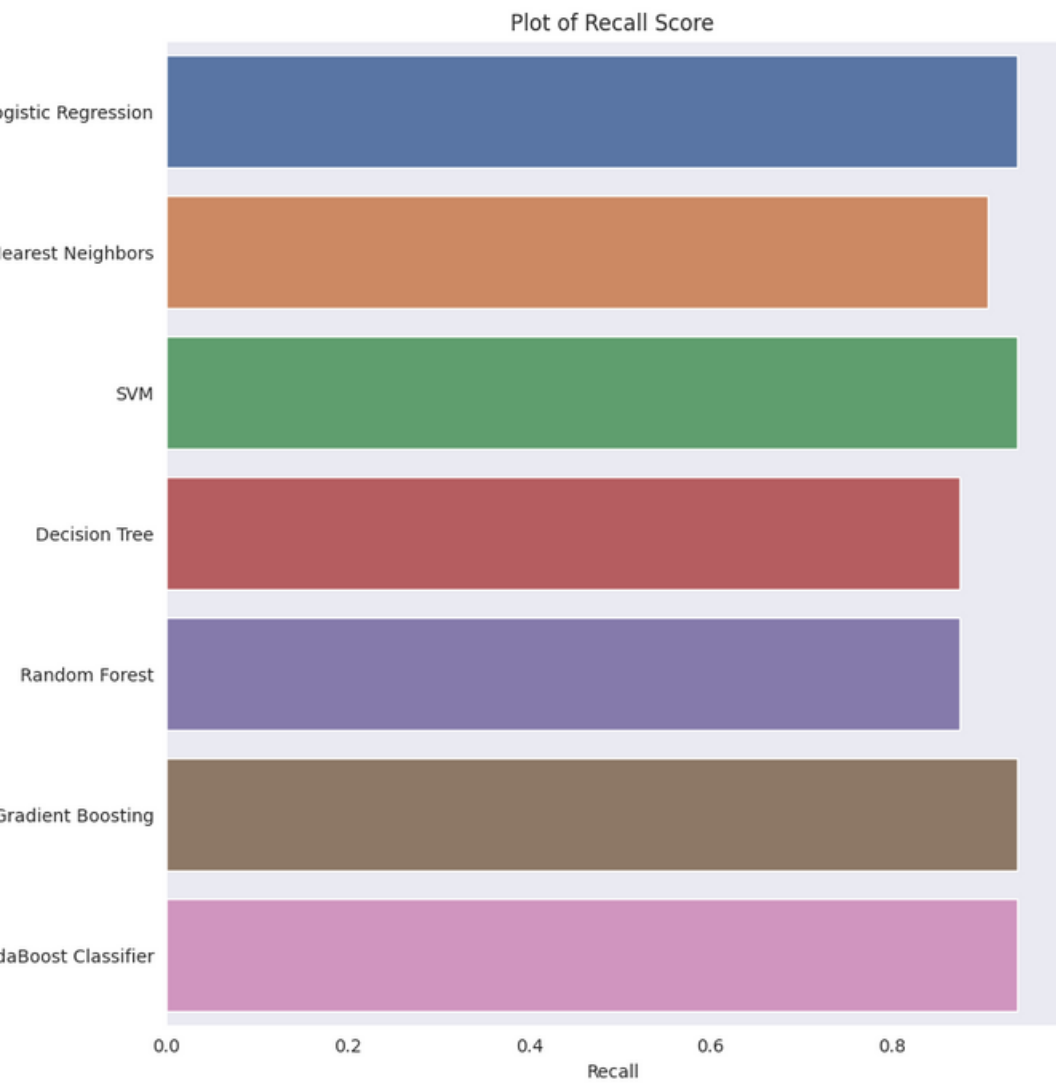
## Gradient Boosting

Accuracy: 0.90164  
Precision: 0.88235  
Recall: 0.9375  
F1: 0.90909

## AdaBoost Classifier

Accuracy: 0.95082  
Precision: 0.96774  
Recall: 0.9375  
F1: 0.95238

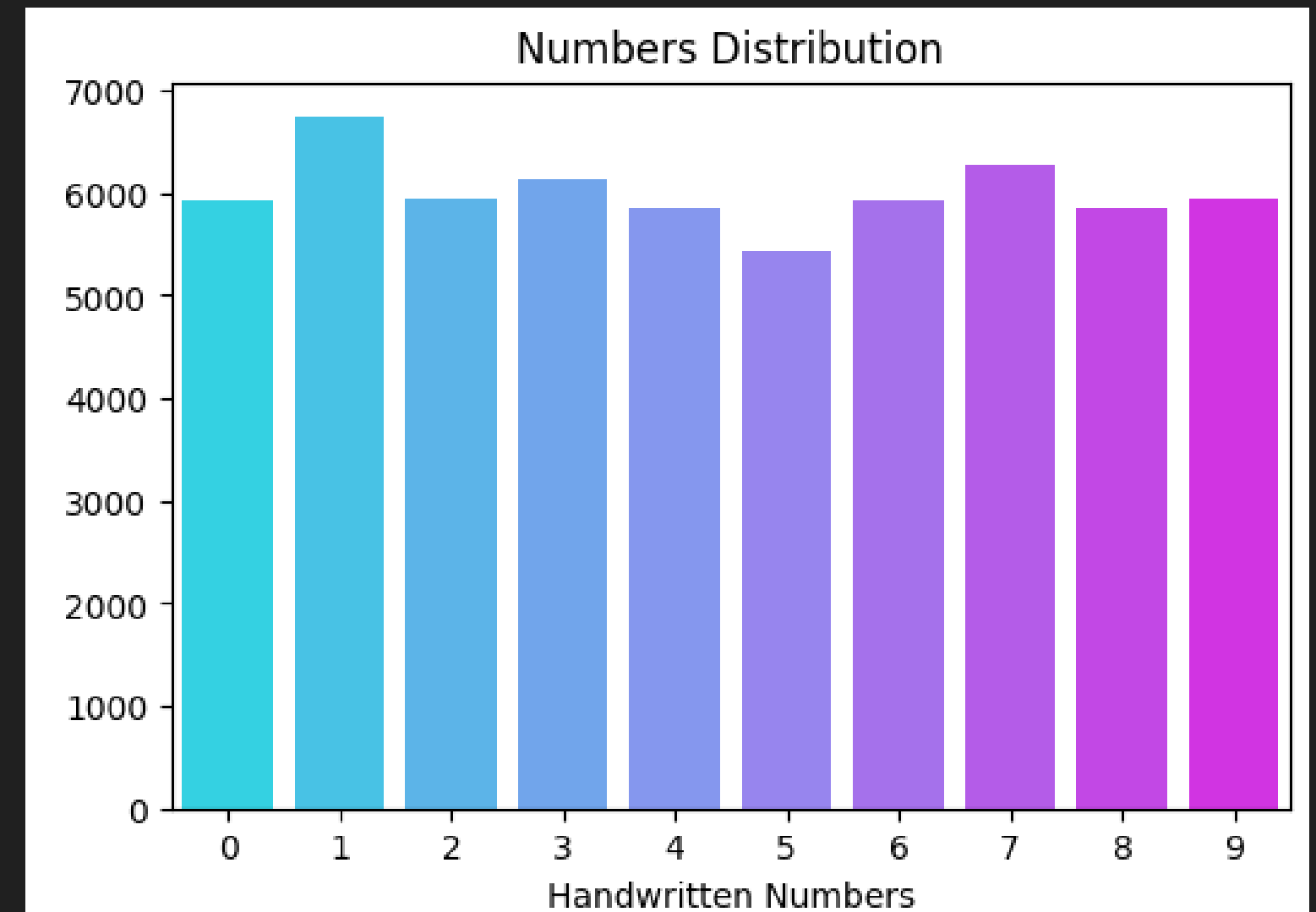
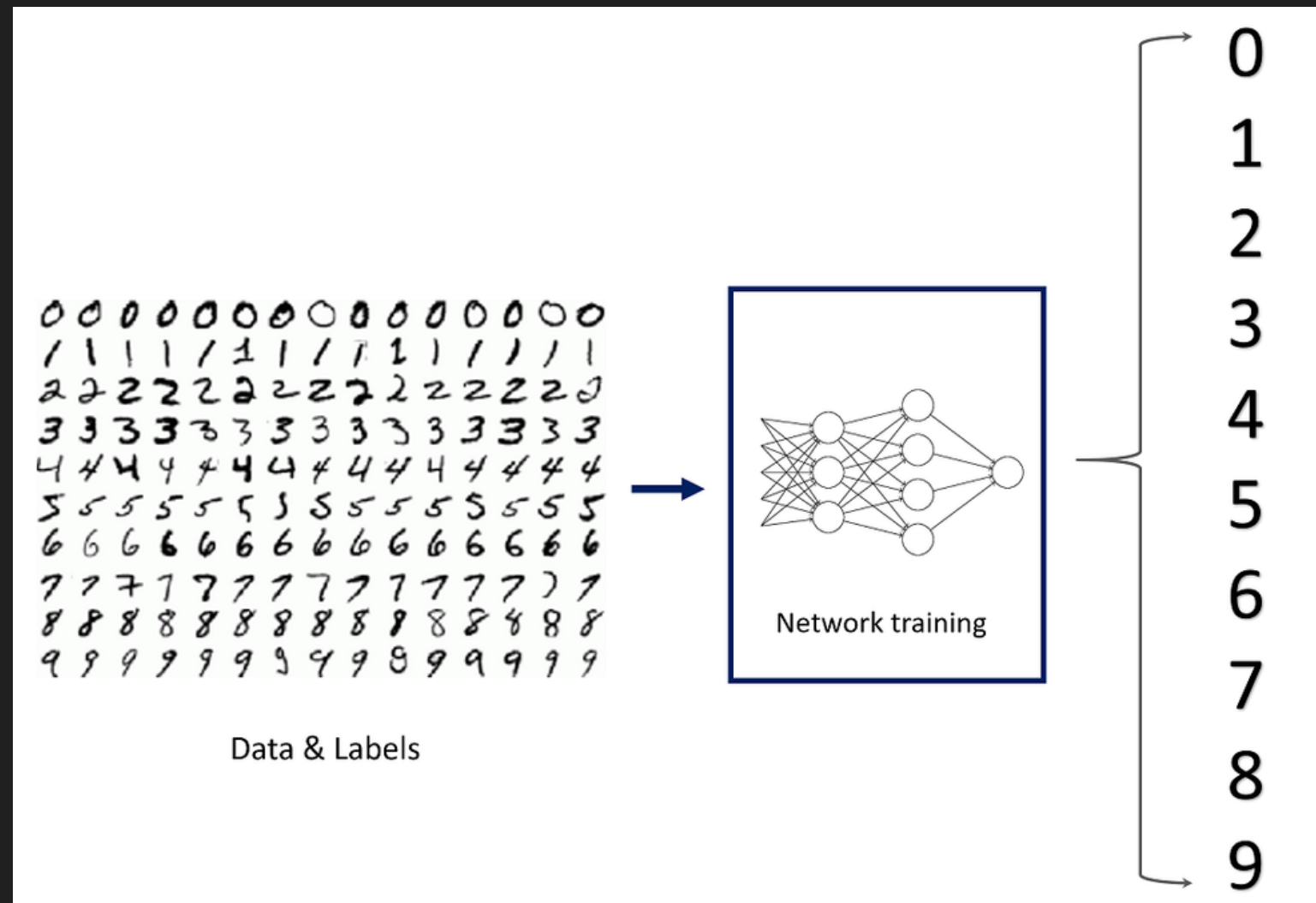


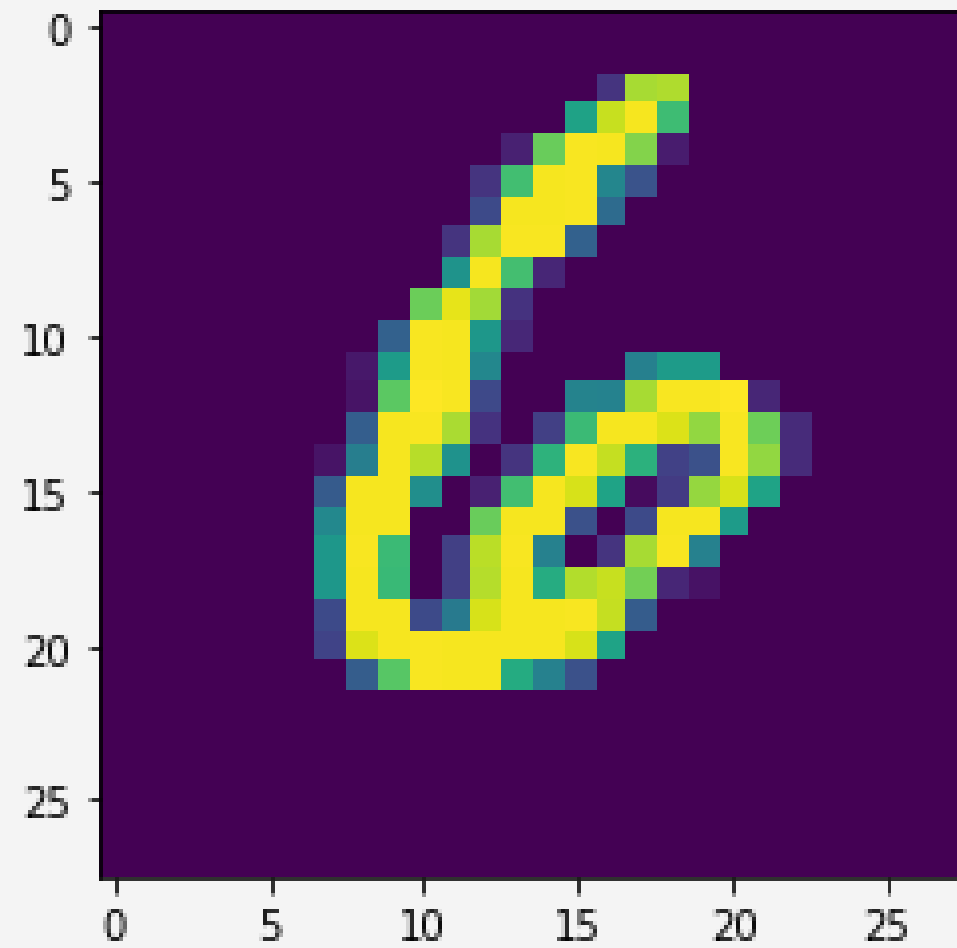


The best ones were ( AdaBoost, SVM and Logistic Regression)

# Image classification - MNIST dataset

60000 observations  
10 labels





visualization of the sample image at index 7777

```
test.shape
(1155, 785)

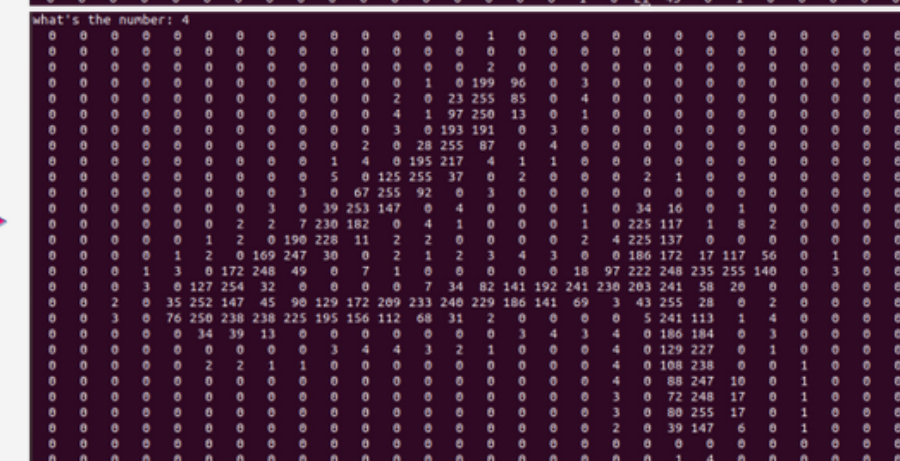
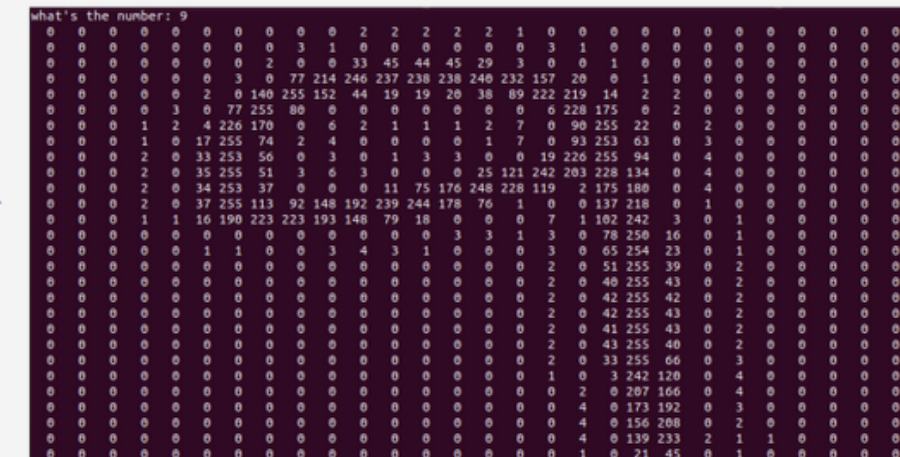
training_dataset_x.shape
(60000, 784)

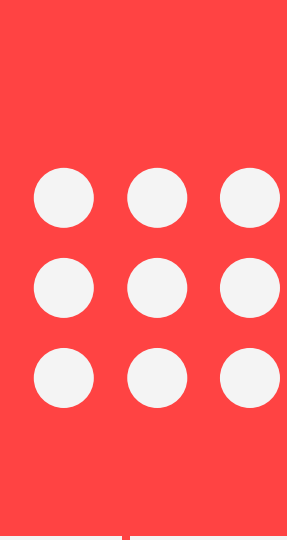
test_dataset_x.shape
(10000, 784)
```

- Reshaping and normalizing the image
- Using scaler transform for SVM

```
X_train = X_train/255
X_test = X_test/255
```

# EDA



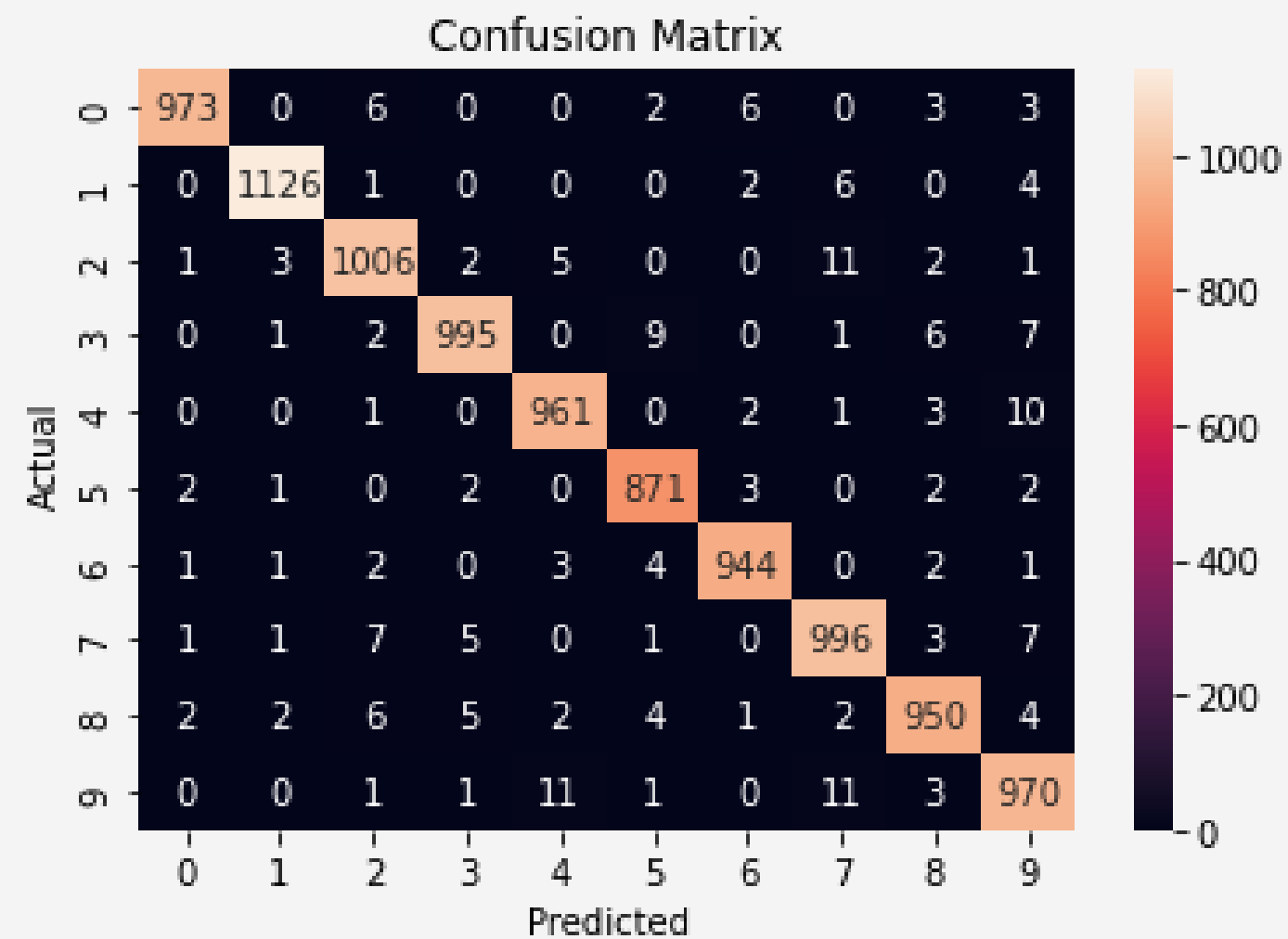


# S V M

## RBF kernel

```
#svm w RBF kernel  
rbf_svm = svm.SVC(kernel='rbf')  
rbf_svm.fit(training_dataset_x, y_train)
```

```
SVC()
```



```
#the accuracy score of the model
```

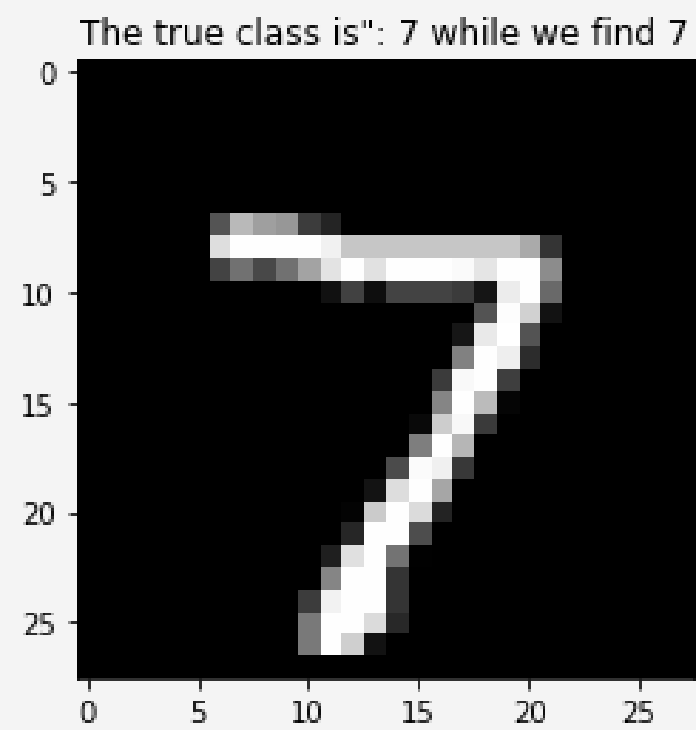
```
accuracy_score(y_pred, y_test)
```

```
0.9792
```

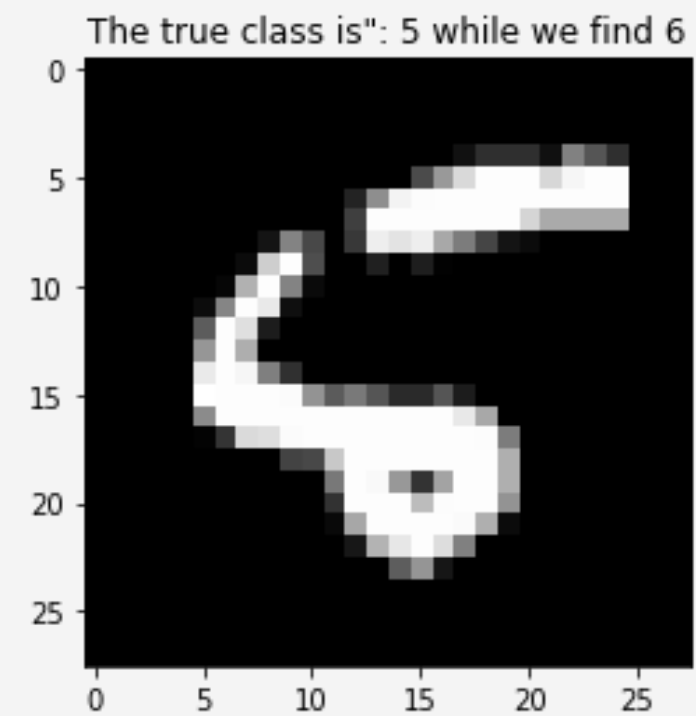




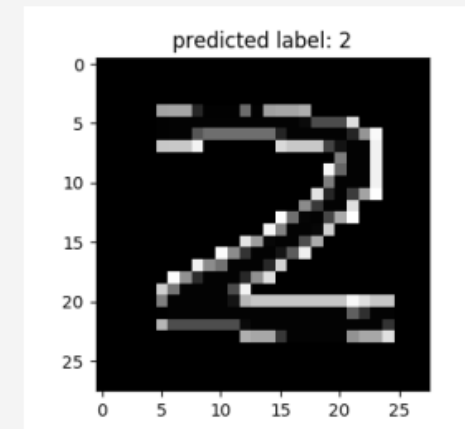
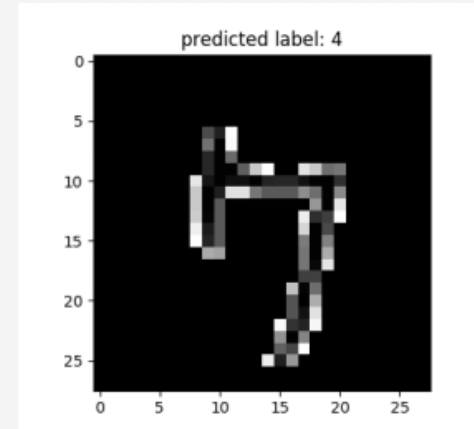
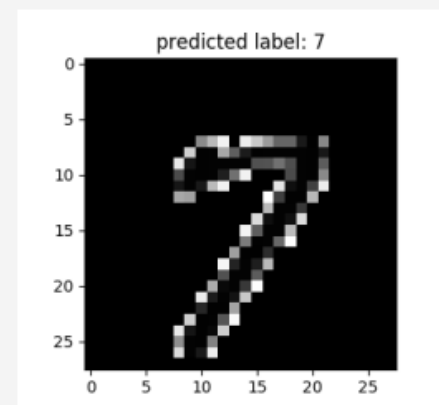
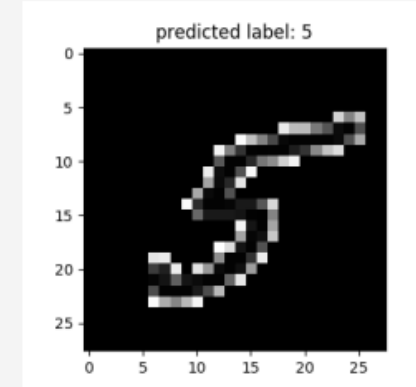
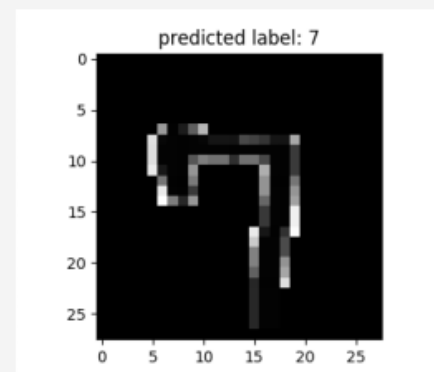
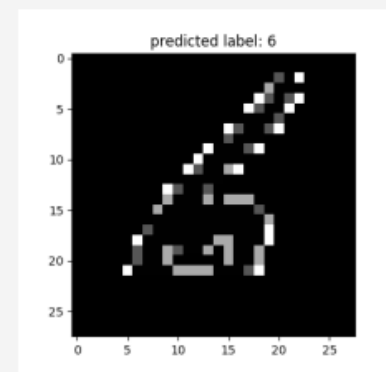
SVM



a case that the model  
correctly classified



a case that the model  
misclassified



# Random Forest

Metrics

Accuracy score -90.6

```
#training random Forest
rf=RandomForestClassifier(n_estimators=100)
rf.fit(training_dataset_x,training_dataset_y)
```

```
RandomForestClassifier()
```

Classification Report					
		precision	recall	f1-score	support
	0	0.99	0.97	0.98	980
	1	1.00	0.98	0.99	1135
	2	0.99	0.89	0.94	1032
	3	1.00	0.87	0.93	1010
	4	0.99	0.89	0.94	982
	5	1.00	0.86	0.92	892
	6	1.00	0.93	0.96	958
	7	0.99	0.91	0.95	1028
	8	0.99	0.83	0.90	974
	9	0.99	0.89	0.93	1009
	micro avg	0.99	0.90	0.95	10000
	macro avg	0.99	0.90	0.94	10000
	weighted avg	0.99	0.90	0.95	10000
	samples avg	0.90	0.90	0.90	10000

# Sequential

Accuracy  
scores for  
train

```
history = model.fit(X_train, y_train, batch_size = 64, epochs = 20, verbose = 1, validation_split = 0.2)
```

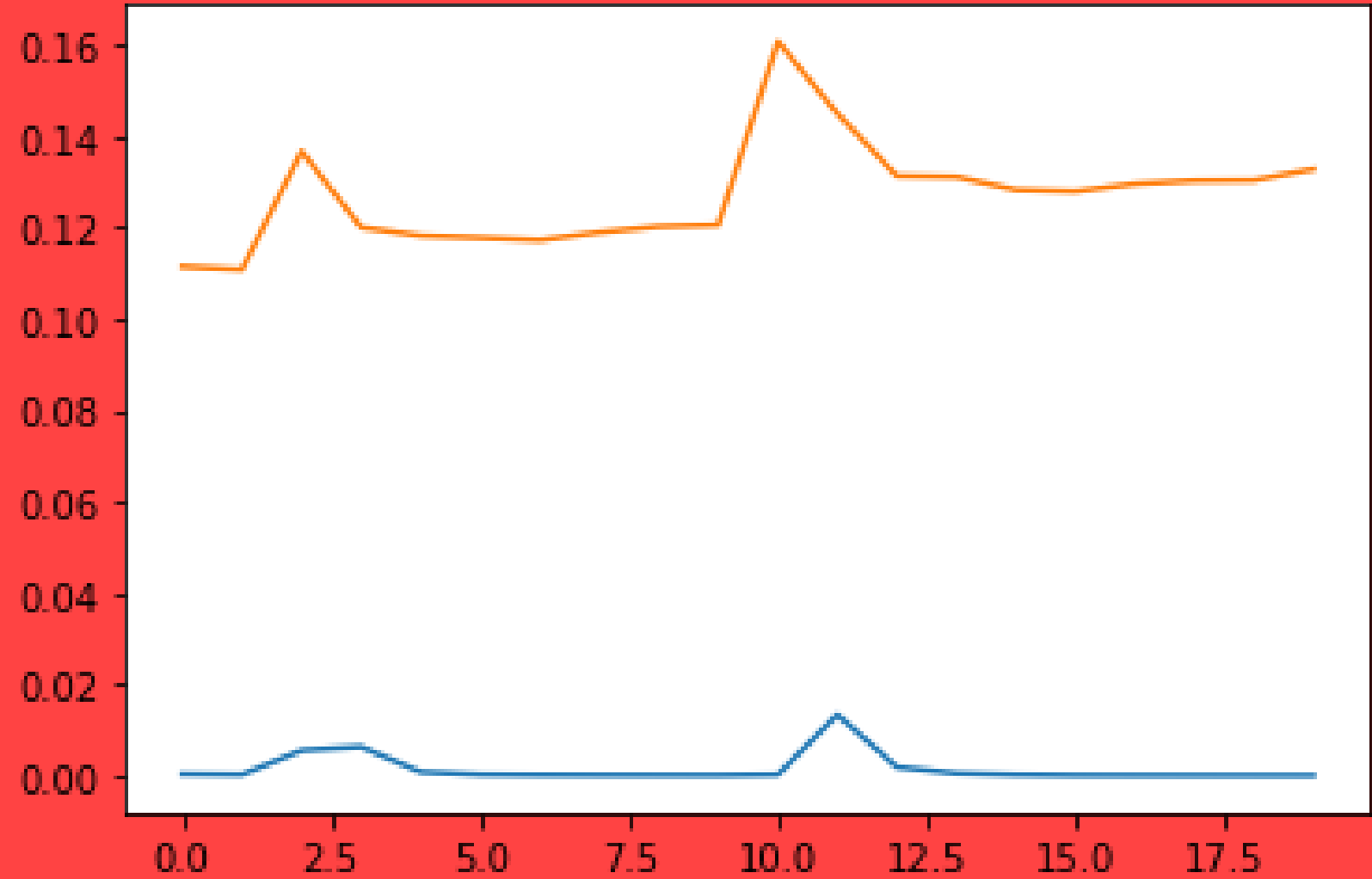
Model: "sequential"

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100480
dense_1 (Dense)	(None, 10)	1290

Total params: 101,770  
Trainable params: 101,770  
Non-trainable params: 0

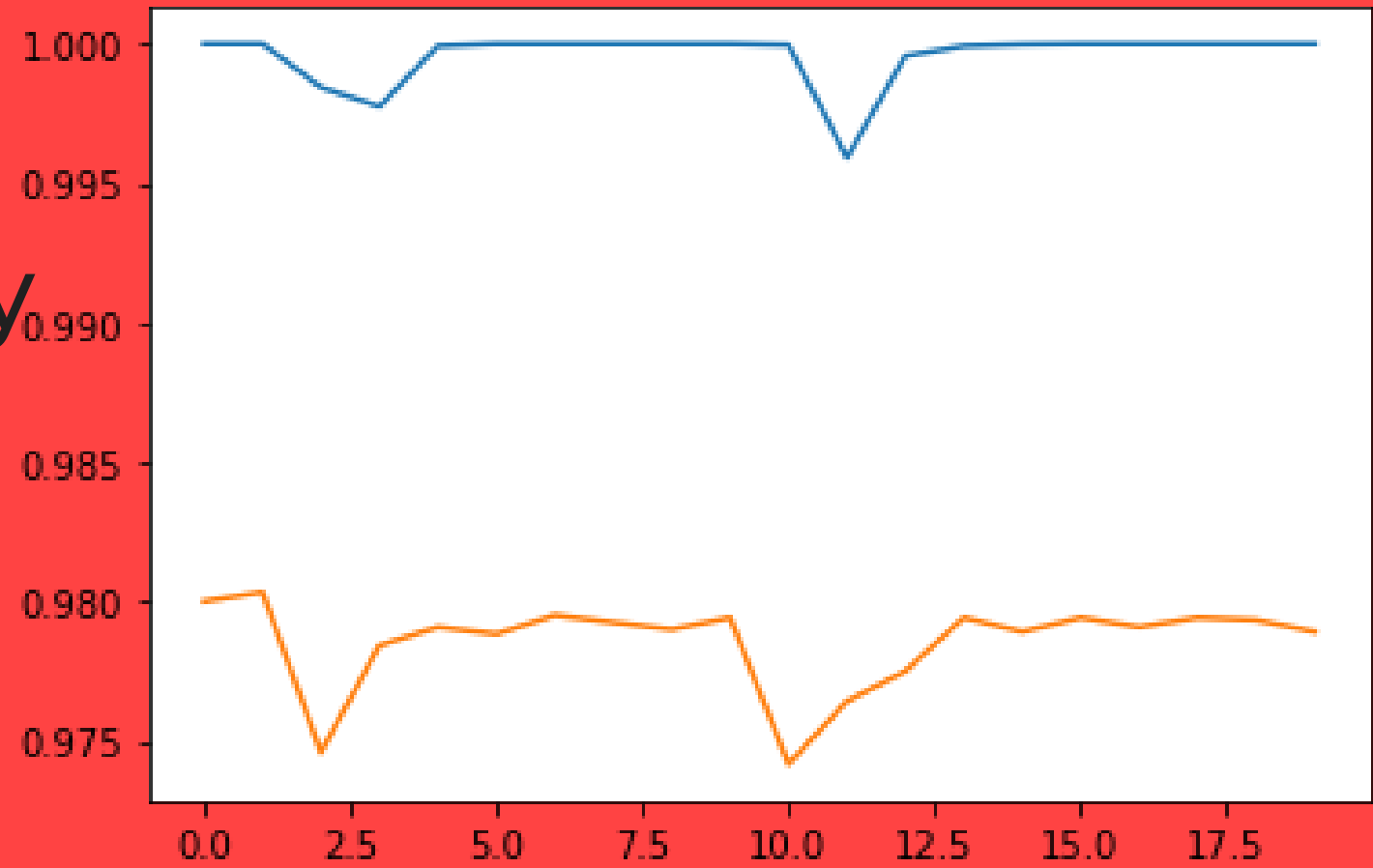


loss vs  
val loss



```
predictions
array([[0, 0, 0, ..., 1, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int32)
```

accuracy vs  
val accuracy



```
2s 2ms/step - loss: 0.0056 - accuracy: 0.9984 - val_loss: 0.1368 - val_accuracy: 0.9746
2s 2ms/step - loss: 0.0064 - accuracy: 0.9977 - val_loss: 0.1201 - val_accuracy: 0.9784
```

example metric

