MRI Detection

PABLO GARCIA
THIBAULT POUX
NABIL KACI

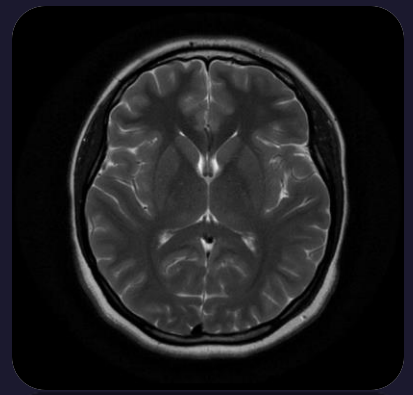
INDEX

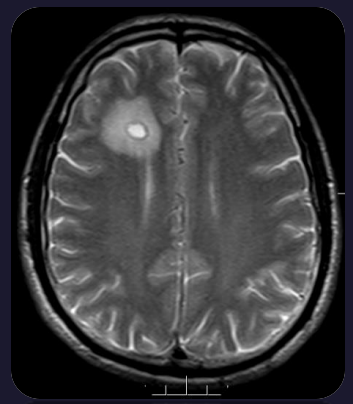
- Brief Explanation in what consists the Project
- Data Processing
- Models classifiers we chose
  - QUADRATIC
  - KNN
  - MLP
  - CNN
  - KNMEANS
- Metrics
  - Roc Curve
  - Confusion Matrix
- Final Explanation on our final pick

MRI-based brain tumor detection

Our Project:

- We're employing machine learning to detect brain tumors in magnetic resonance imaging (MRI) scans.
- Our process involves data preprocessing and building several model
- Our aim is to develop accurate models to aid in early diagnosis and effective treatment of brain tumors.
- The general idea for each problem has been to test several classifiers and choose the best one's according their accuracy and after those three we have carried out several tests to see which one is the best for each problem



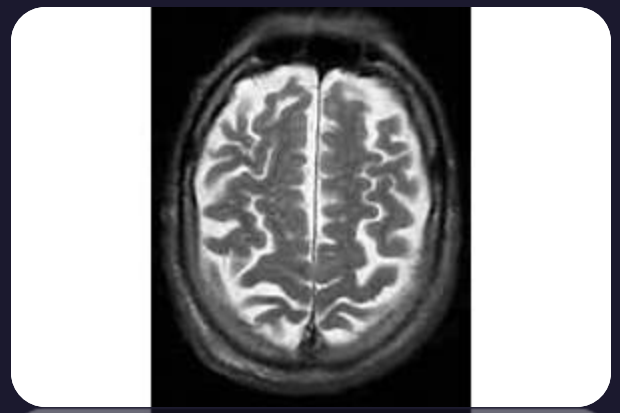


DATA IMBALANCE:

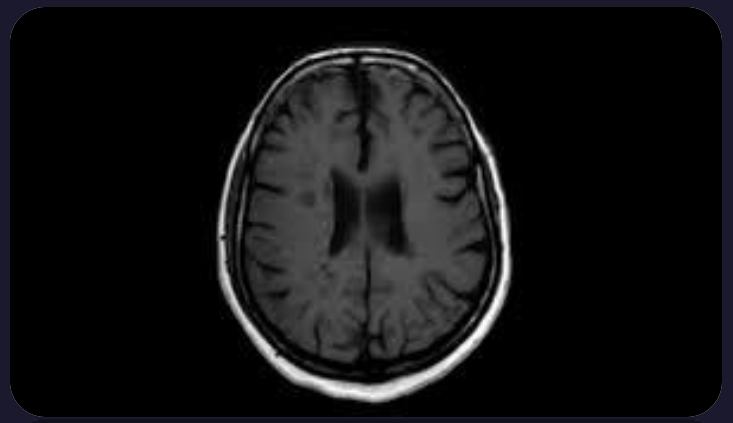
- Images with brain tumors: 155
- Images with no brain tumors: 98

Data Processing

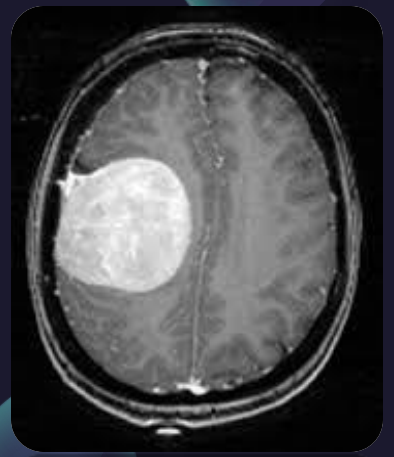
- For our project, we chose a dataset from Kaggle. However, as you can see, most of the images have different shapes, sizes, and backgrounds.



Size 300*200



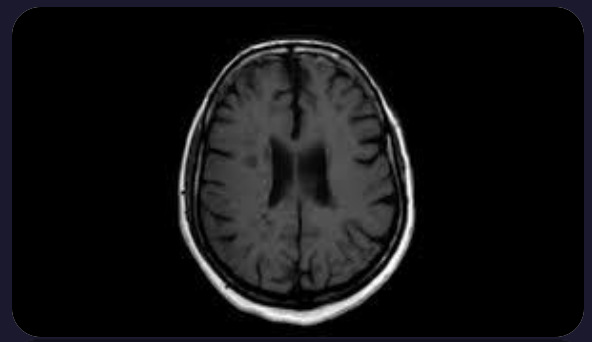
Size 300*168

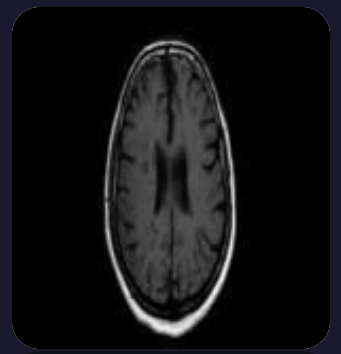


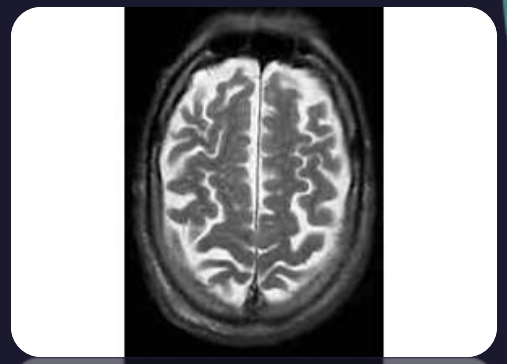
Size 180*218

Data Processing

- The first thing we tried was resizing the image, but it's clear that all the images are now distorted.







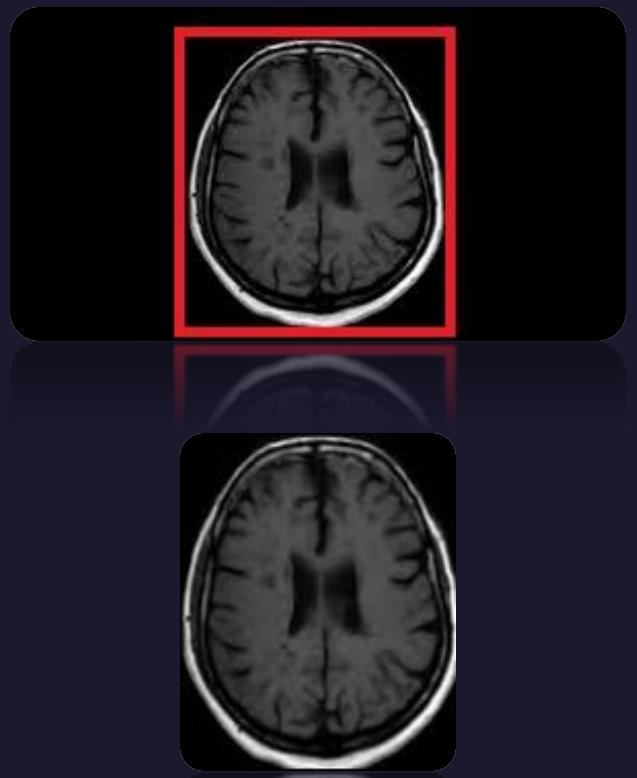
Data Processing

- Our next approach was to use a Sobel filter to detect the outline of the brain and resize the image accordingly, aiming to remove all unnecessary background.

```python
# load the image in gray scale
image = cv2.imread(file_name, cv2.IMREAD_GRAYSCALE)

# Convolution Matrix (Sobel Filter)
kernel_x = np.array([[-1, 0, 1],
                     [-2, 0, 2],
                     [-1, 0, 1]])

kernel_y = np.array([[-1, -2, -1],
                     [0, 0, 0],
                     [1, 2, 1]])


# aply the filter to the image
edges_x = cv2.filter2D(image, -1, kernel_x)
edges_y = cv2.filter2D(image, -1, kernel_y)

# combined the axis
edges = cv2.addWeighted(edges_x, 0.5, edges_y, 0.5, 0)

# only keep the edges that are above a certain threshold
_, thresholded = cv2.threshold(edges, 100, 255, cv2.THRESH_BINARY)

# find all the contours taht can be found in the image
contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Data Processing

- Our next approach was to use a Sobel filter to detect the outline of the brain and resize the image accordingly, aiming to remove all unnecessary background.



Models

QUADRATIC MODEL

- A quadratic model is a mathematical representation that includes terms with variables raised to the second power. It describes a non-linear relationship between variables, often represented by a curve rather than a straight line.

KNN

- KNN, or K-Nearest Neighbors, is a simple yet powerful algorithm used for classification and regression tasks. In KNN, when we want to make a prediction for a new data point, we look at the 'K' nearest data points to it (nearest neighbors) based on some similarity measure (usually Euclidean distance). Then, for classification, we assign the majority class among these neighbors to the new data point, and for regression, we take the average of the target values of these neighbors. It's like asking your friends who live closest to you to help you make a decision.

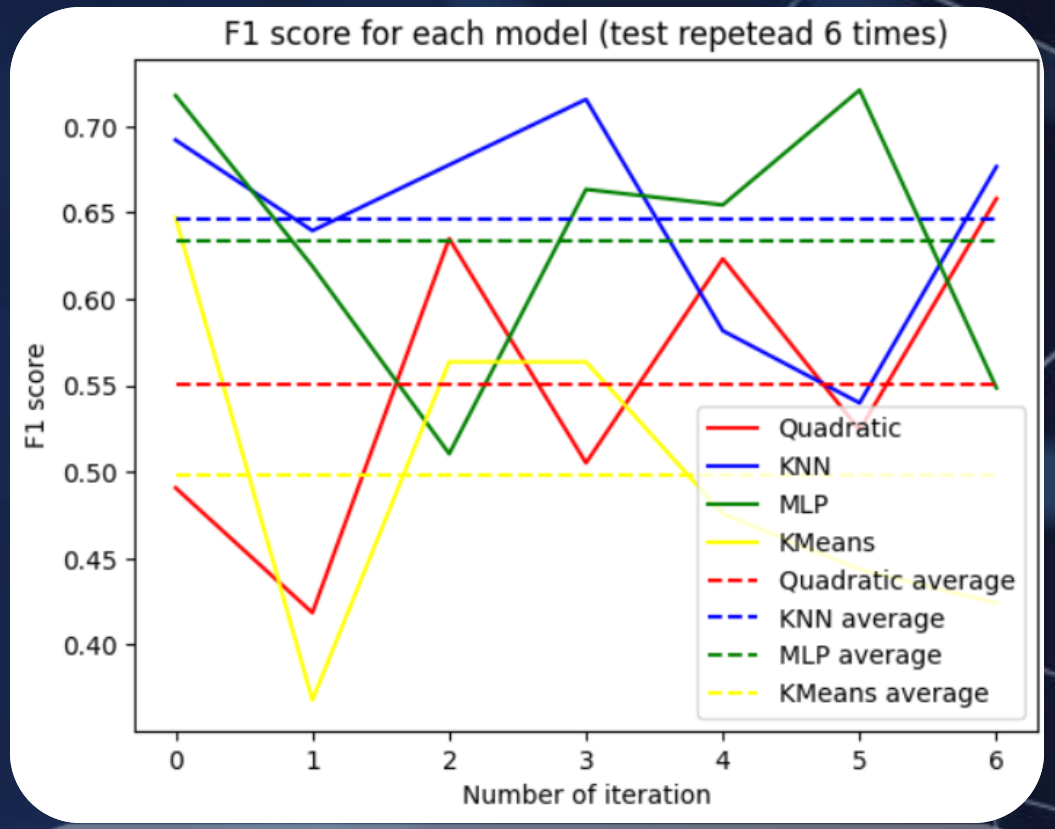
MLP

- MLP, or Multi-Layer Perceptron, is a type of neural network with layers of interconnected nodes. It's used for tasks like classification and regression. Information flows through the network, and each layer applies transformations to the data. Training involves adjusting weights to minimize prediction errors.

KMEANS

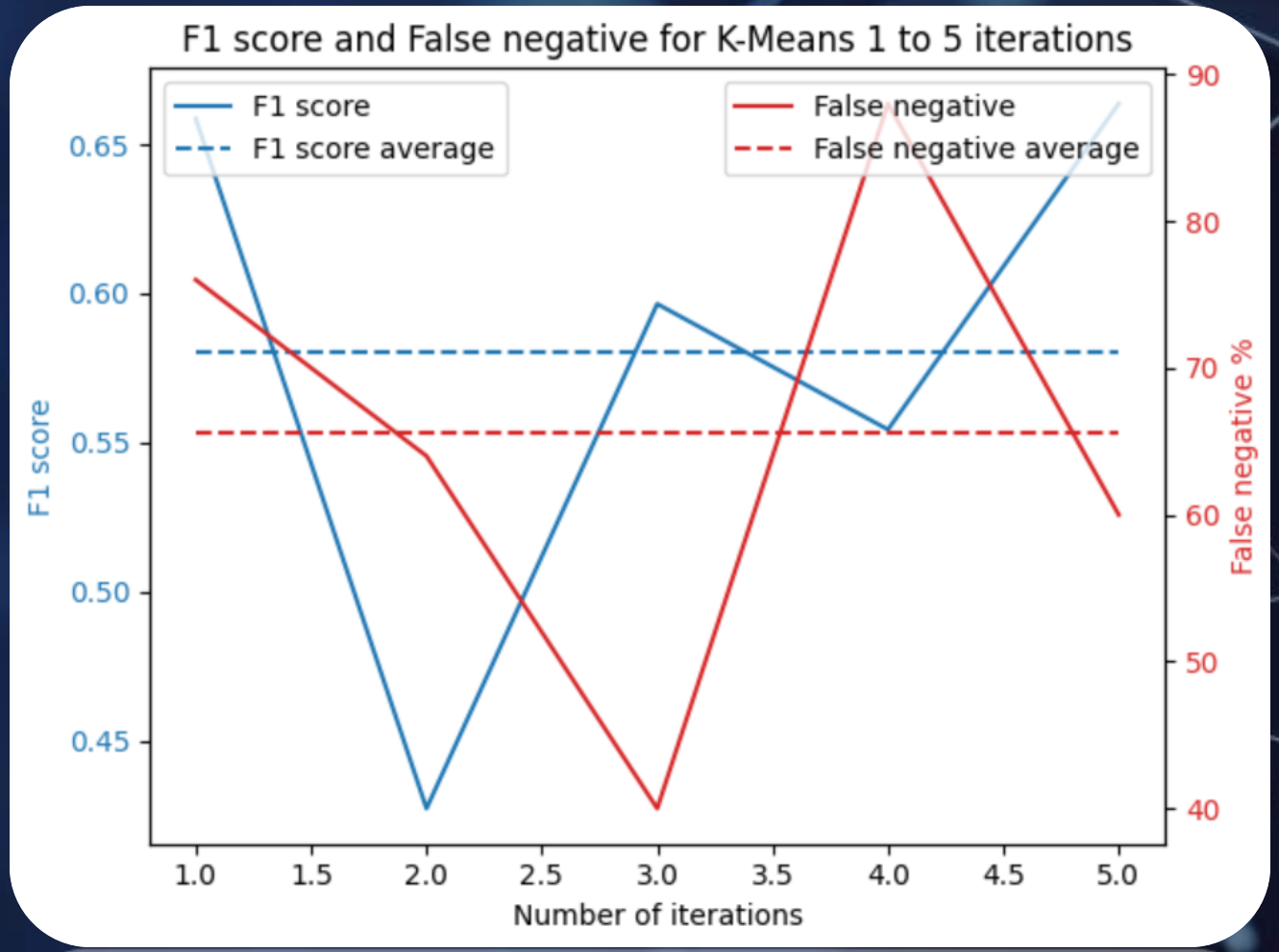
- K-means is a clustering algorithm that groups data points into K clusters based on their similarity. It works by iteratively updating cluster centroids until convergence. Simple and efficient, it's used for tasks like customer segmentation and image compression.

GENERAL OVERVIEW OF MODELS

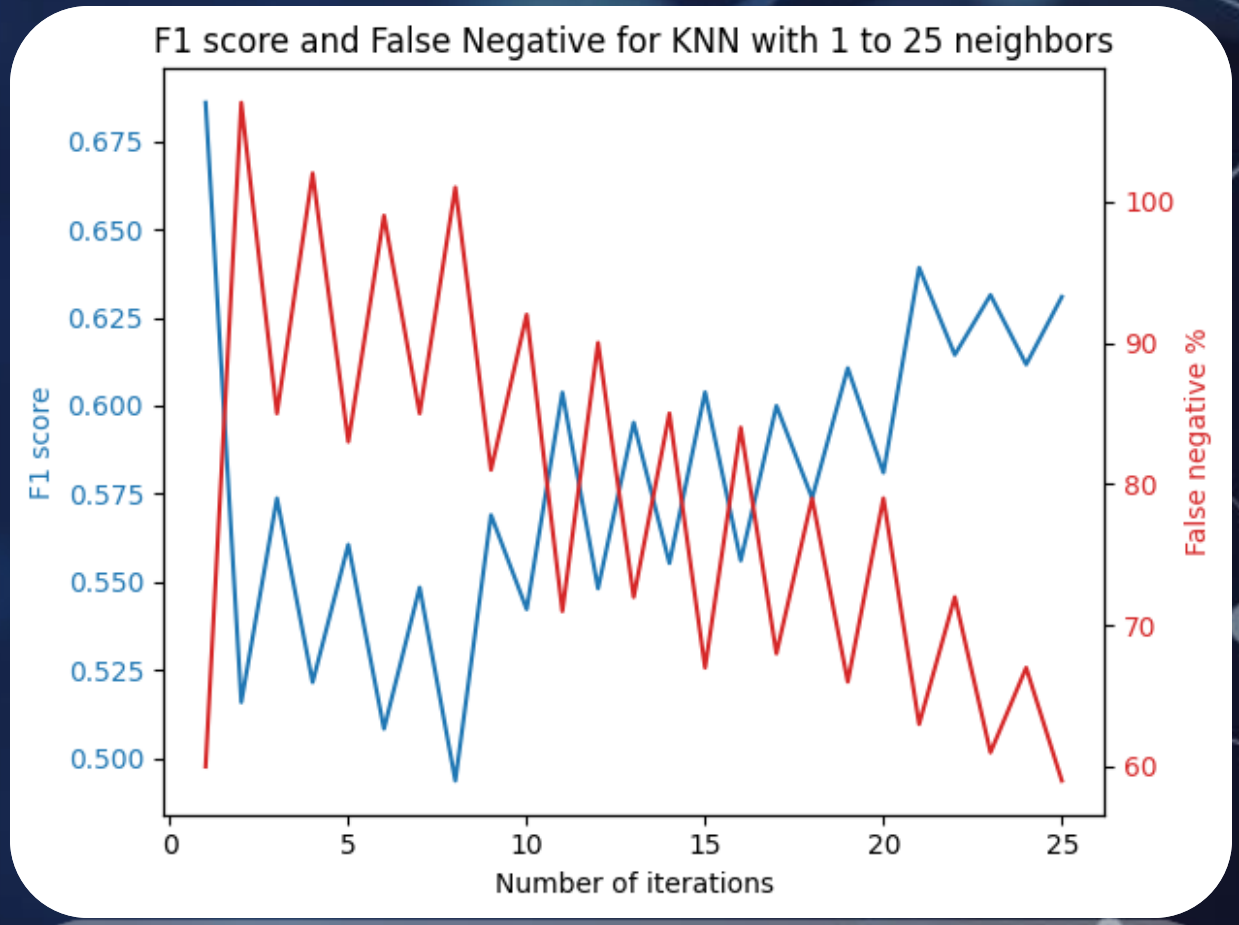


Model	F1 Score	Accuracy	False negative	Time to fit	Time to predict
Quadratic Discriminant Analysis	0.55	0.61	34.69%	0.544024	0.0391852
K-Nearest Neighbors	0.65	0.68	28.74%	0.000734295	0.043933
Multi-Layer Perceptron	0.63	0.62	13.44%	4.65964	0.0089252
KMeans	0.5	0.49	28.23%	0.123986	0.0110728

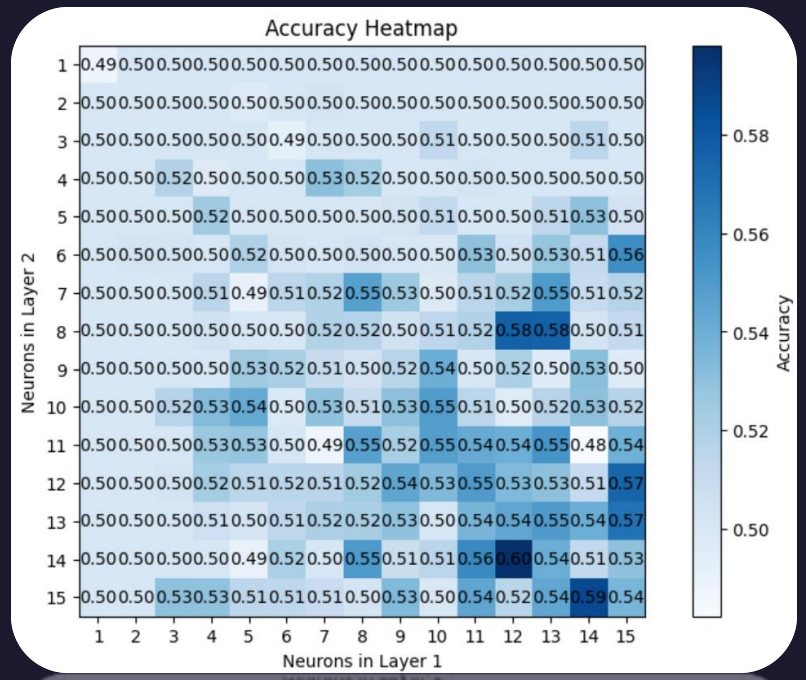
KMEANS

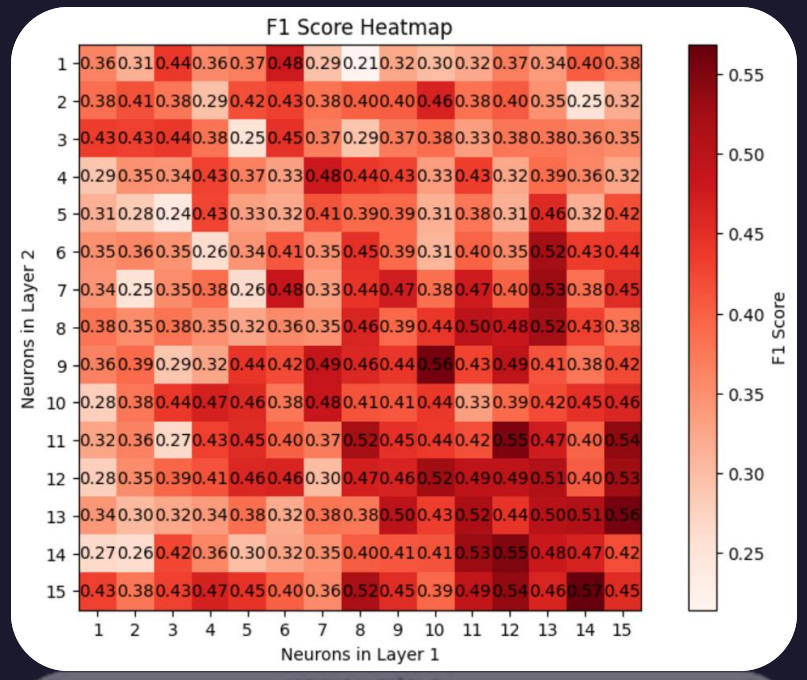

F1 score and False negative for K-Means 1 to 5 iterations
F1 score
F1 score average
False negative
False negative average
F1 score
False negative %
Number of iterations
0.65
0.60
0.55
0.50
0.45
90
80
70
60
50
40
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

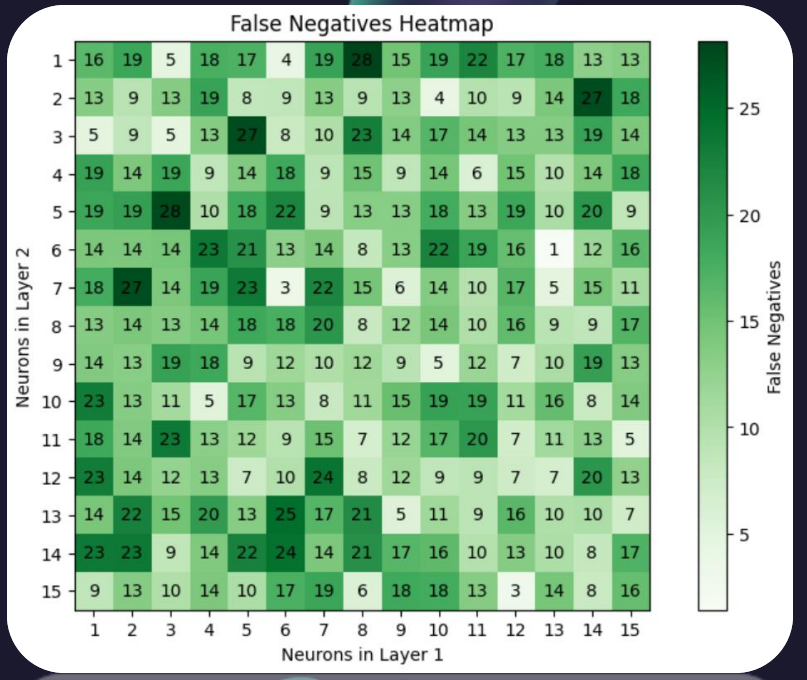
KNN


F1 score and False Negative for KNN with 1 to 25 neighbors
F1 score
False negative %
Number of iterations
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
100
90
80
70
60
0
5
10
15
20
25

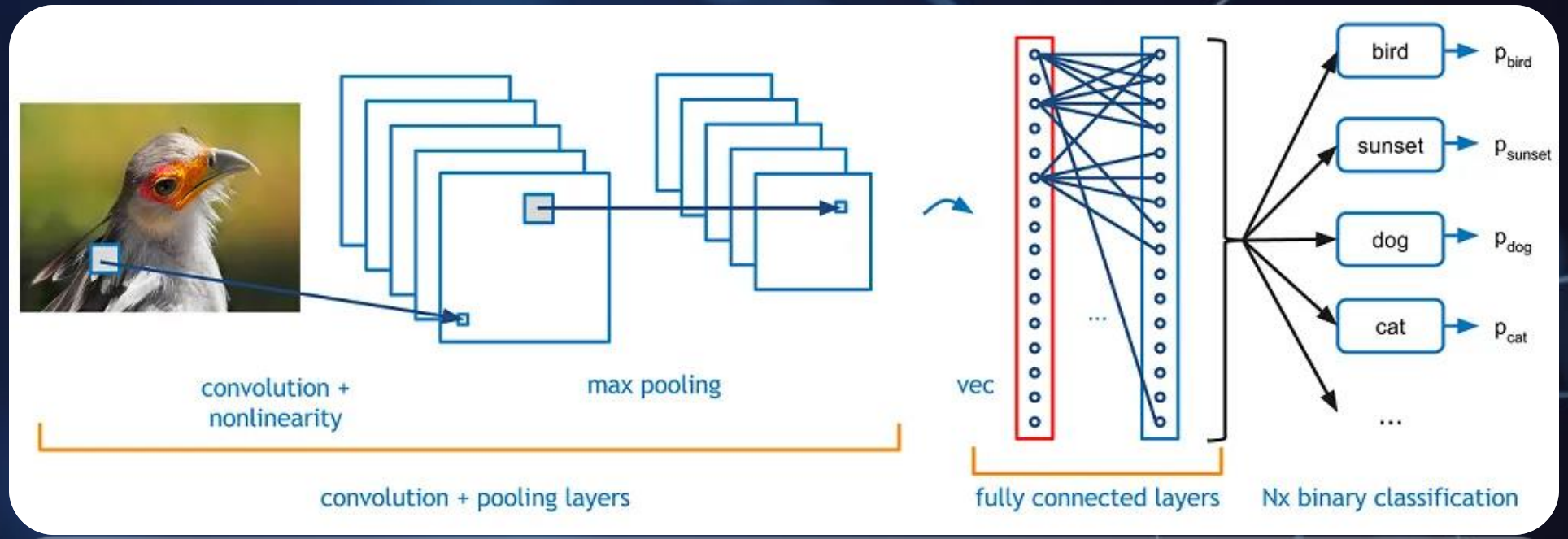
MLP


Accuracy Heatmap
Neurons in Layer 2
Neurons in Layer 1
Accuracy
0.58
0.56
0.54
0.52
0.50


F1 Score Heatmap
Neurons in Layer 2
Neurons in Layer 1
F1 Score
0.55
0.50
0.45
0.40
0.35
0.30
0.25


False Negatives Heatmap
Neurons in Layer 2
Neurons in Layer 1
False Negatives
25
20
15
10
5

CNN


convolution +
nonlinearity
max pooling
vec
bird
sunset
dog
cat
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification

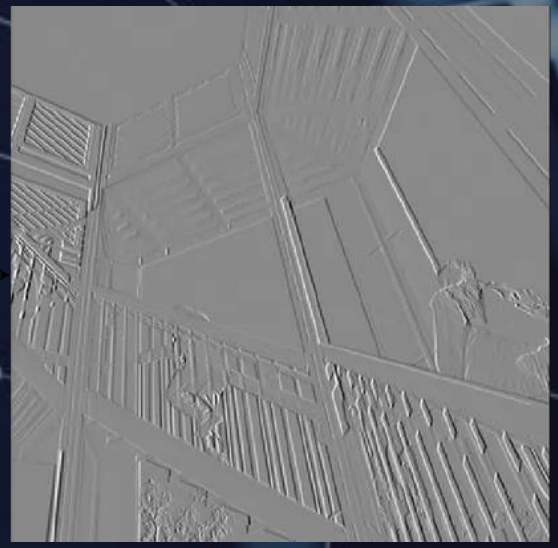
Convolution Layer

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

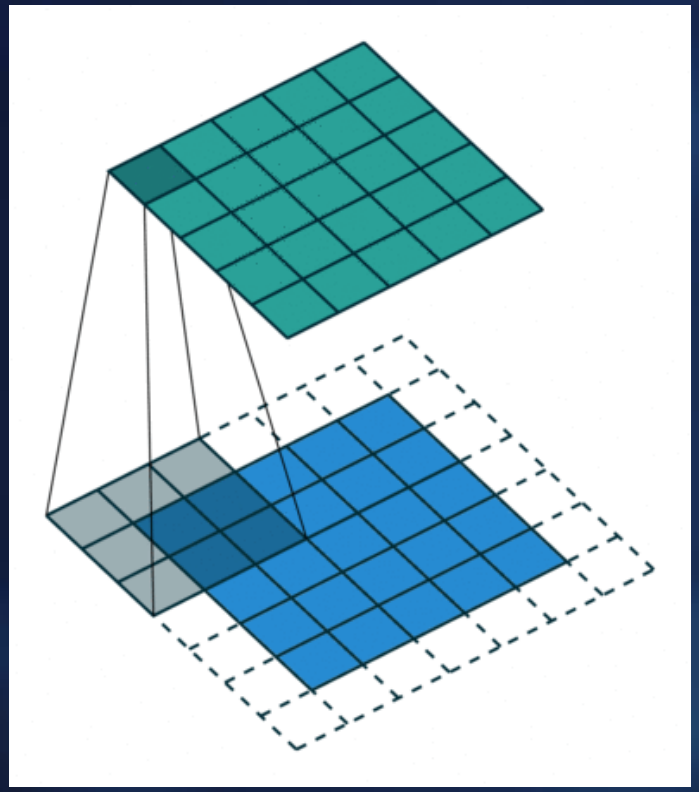
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

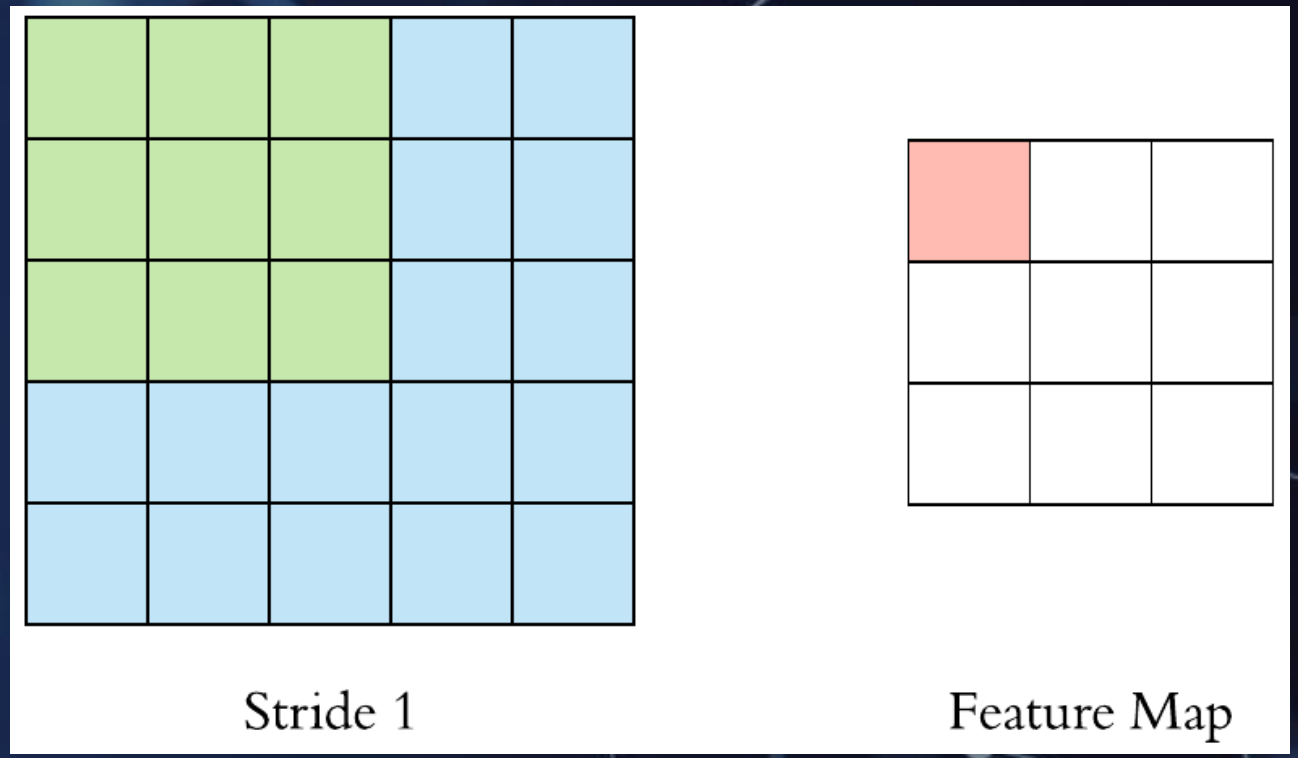
Horizontal Sobel kernel



Convolution Layer

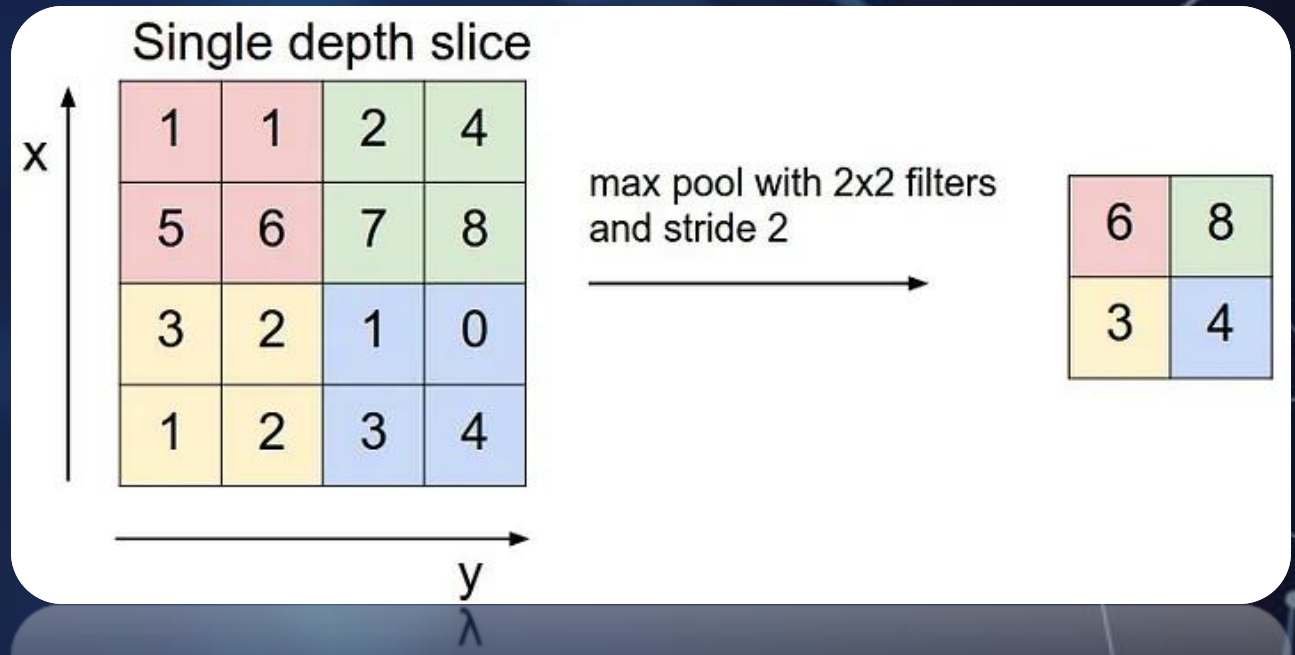


Padding method

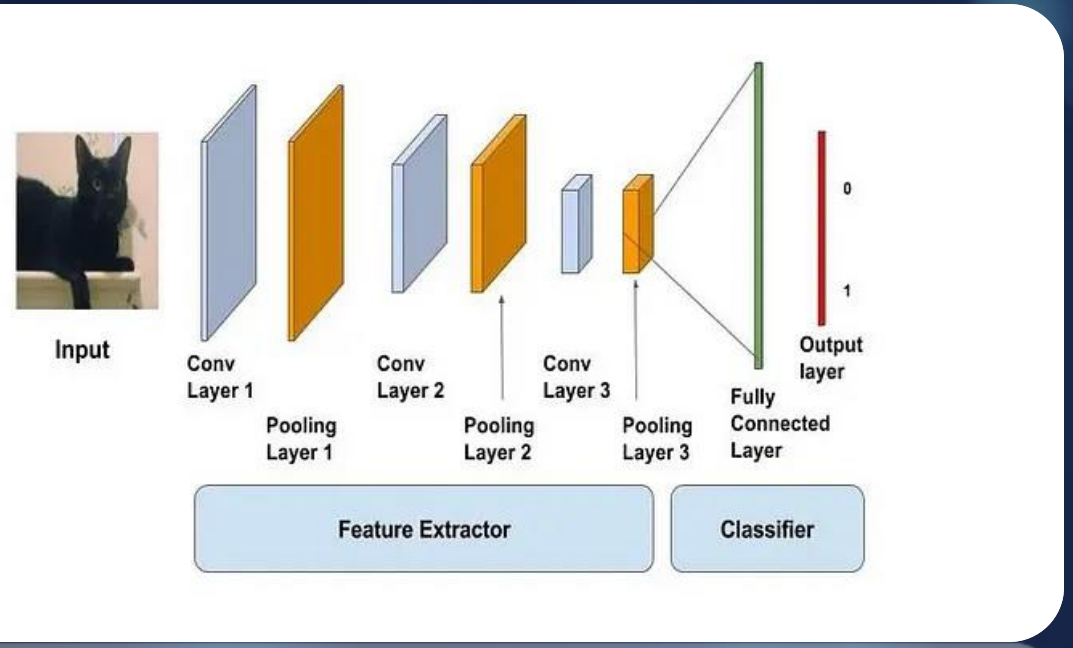


Stradding method

Pooling Layer



CNN



Layer (type)	Output Shape	Param #
conv2d_40 (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d_40 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_41 (Conv2D)	(None, 61, 61, 32)	9,248
max_pooling2d_41 (MaxPooling2D)	(None, 30, 30, 32)	0
flatten_20 (Flatten)	(None, 28800)	0
dense_39 (Dense)	(None, 65)	1,872,065
dense_40 (Dense)	(None, 1)	66

Total params: 1,882,275 (7.18 MB)

Trainable params: 1,882,275 (7.18 MB)

Non-trainable params: 0 (0.00 B)

CNN

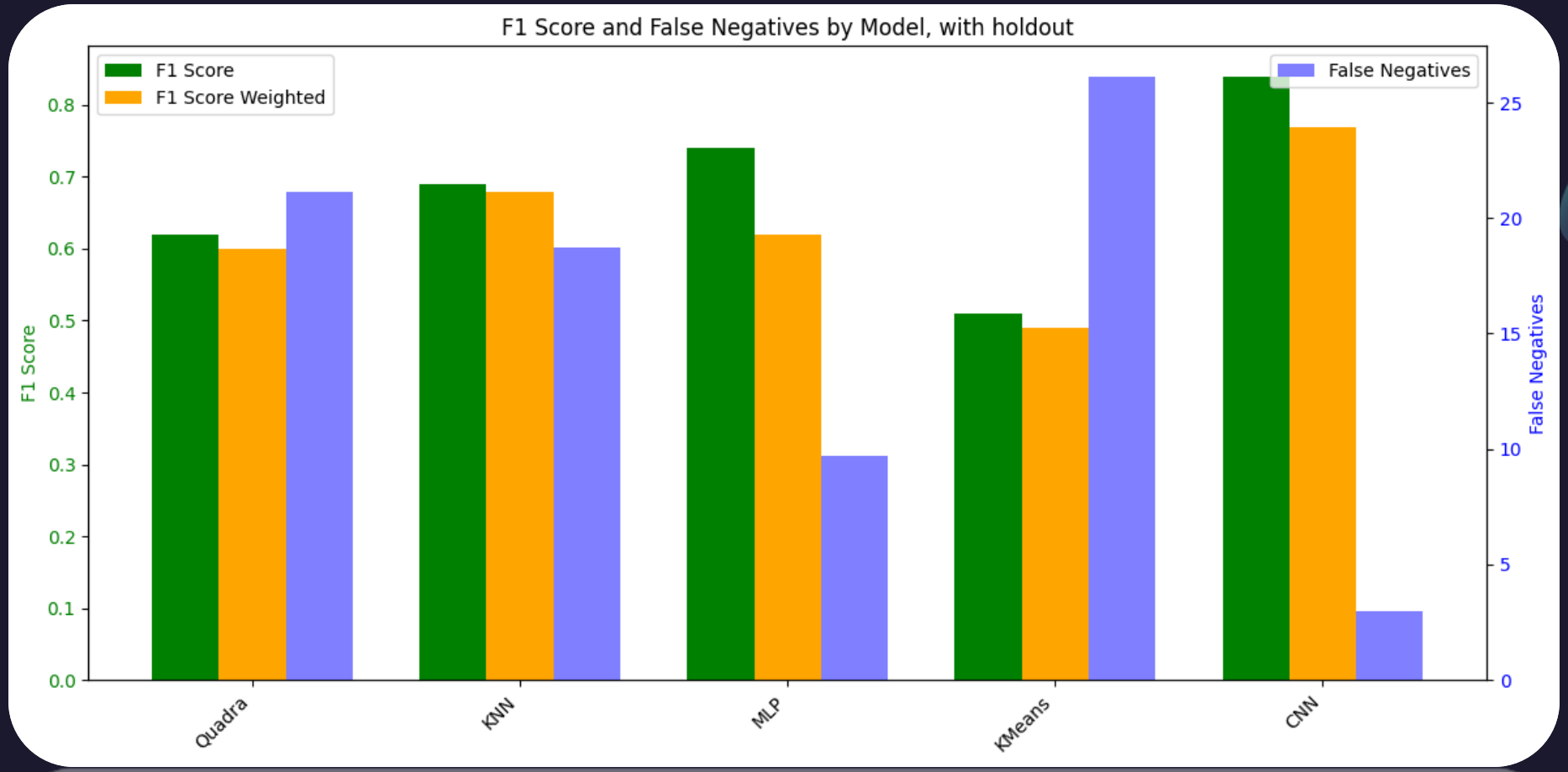
```
# Define F1 score metric
f1 = F1Score(average=None, threshold=None, name="f1_score", dtype=None)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics= ['accuracy',f1])   # Compile
model.fit(X_train,y_train, epochs= 10, batch_size=64 ,class_weight={0: 155/253, 1: 98/253})
```

```
Epoch 1/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 81ms/step - accuracy: 0.4999 - f1_score: 0.6732 - loss: 30.1679
Epoch 2/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.5284 - f1_score: 0.7846 - loss: 4.7435
Epoch 3/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.6885 - f1_score: 0.7667 - loss: 0.7210
Epoch 4/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.6677 - f1_score: 0.7599 - loss: 0.5471
Epoch 5/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - accuracy: 0.8343 - f1_score: 0.7476 - loss: 0.1899
Epoch 6/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - accuracy: 0.8565 - f1_score: 0.7768 - loss: 0.1262
Epoch 7/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9374 - f1_score: 0.7428 - loss: 0.0791
Epoch 8/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9755 - f1_score: 0.7507 - loss: 0.0522
Epoch 9/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.9631 - f1_score: 0.7581 - loss: 0.0391
Epoch 10/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 1.0000 - f1_score: 0.7719 - loss: 0.0118
```

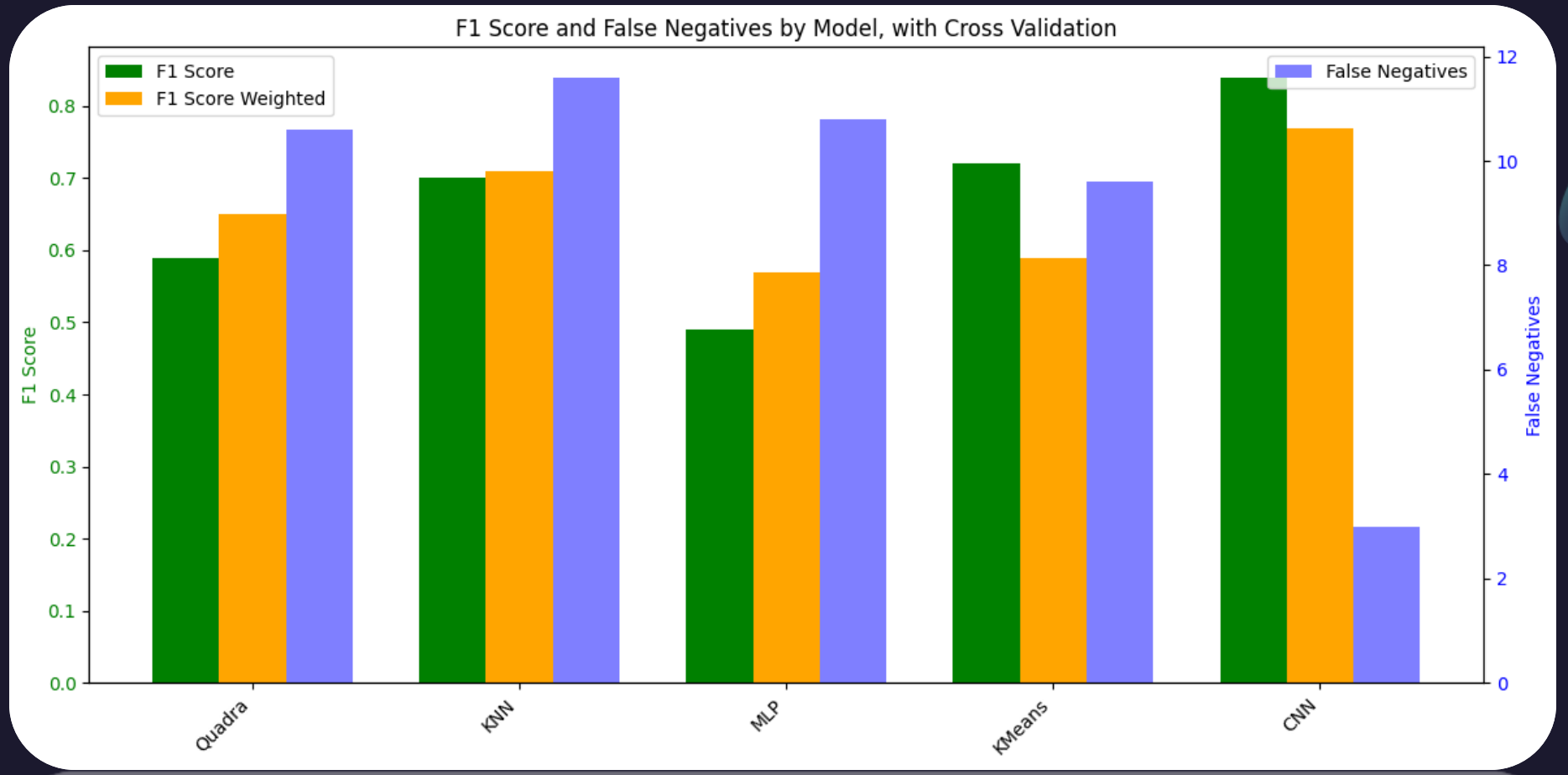
```
model.evaluate(X_test, y_test)      # Evaluate the trained model on X_test and y_test data
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.8303 - f1_score: 0.7782 - loss: 1.0243

[1.050042748451233, 0.8235294222831726, 0.7710843086242676]
```

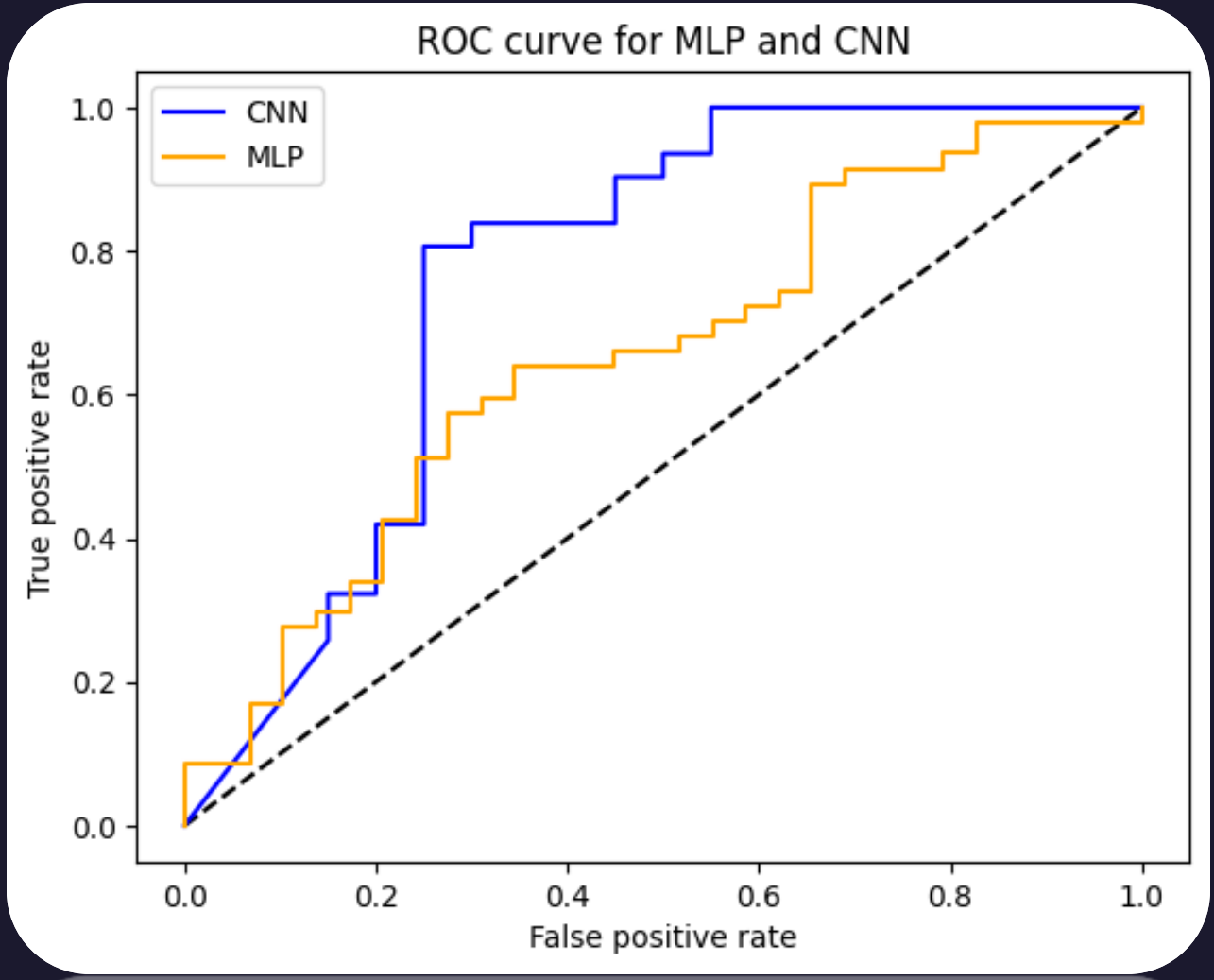
Conclusion


F1 Score and False Negatives by Model, with holdout
F1 Score
F1 Score Weighted
False Negatives
F1 Score
False Negatives
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
Quadra
KNN
MLP
KMeans
CNN

Conclusion


F1 Score and False Negatives by Model, with Cross Validation
F1 Score
F1 Score Weighted
False Negatives
F1 Score
False Negatives
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
2
4
6
8
10
12
Quadra
KNN
MLP
KMeans
CNN

Conclusion


ROC curve for MLP and CNN
CNN
MLP
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

Thanks
Any Questions?

PABLO GARCIA

THIBAULT POUX

NABIL KACI