Chest X-Ray Pneumonia Classification

By: Kasey Larsen

Introduction of Data

Chest x-rays are analyzed to look for the presents of pneumonia

Our Data Set:

5840 Chest X-ray images

Labeled “Normal” or “Pneumonia”

Problem to Solve

Use the chest x-rays to make classifications

Limit misclassified chest x-rays

Metrics for Measure:

Accuracy, Precision, **Recall** and f1 scores

Confusion Matrix

Data Understanding and Cleaning

THERE WAS NOT MUCH CLEANING NECESSARY

ALL IMAGES INCLUDED LABELS

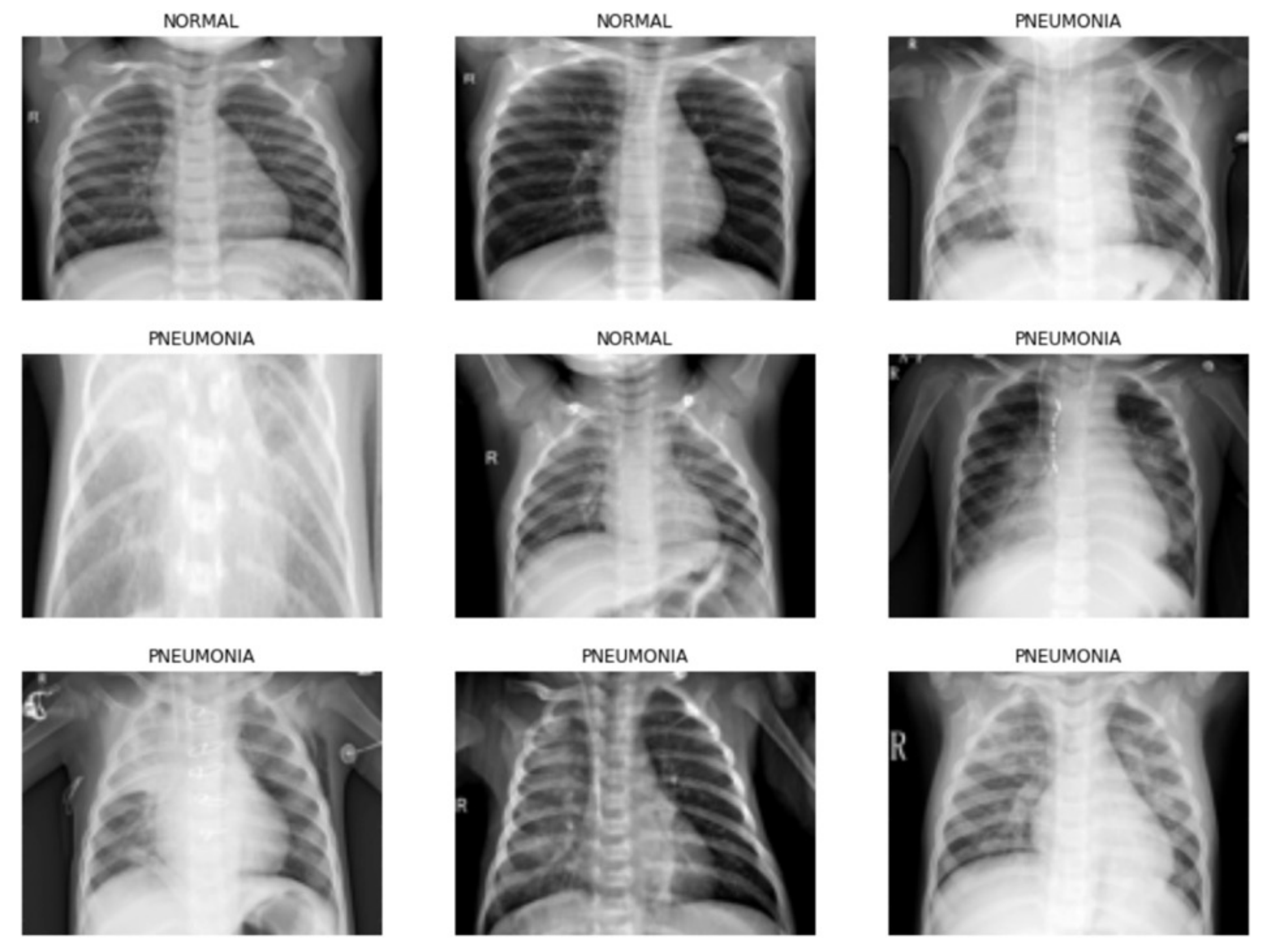
IMAGES WERE RESIZED TO 132 X 97

Exploratory Data Analysis Step 1

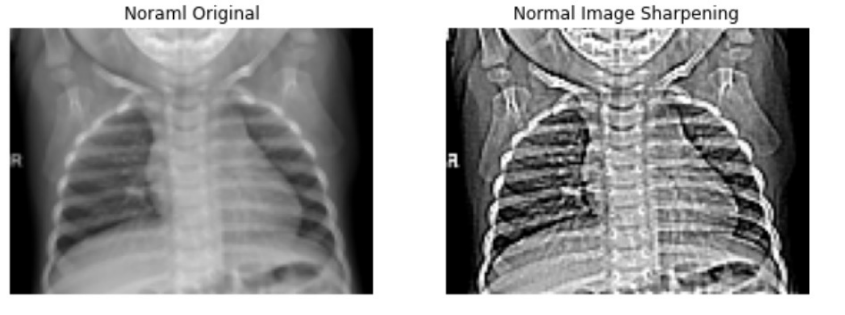
4,265 out of 5840 were labeled “pneumonia”

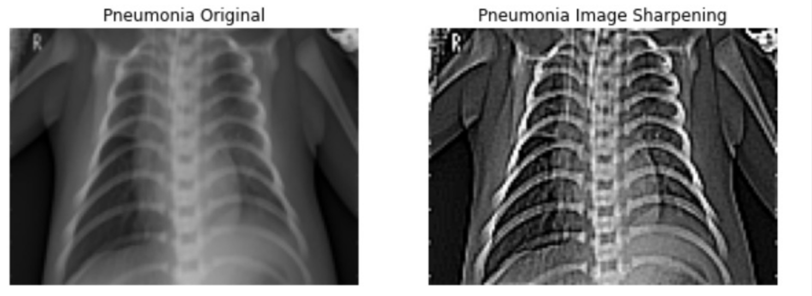
1,575 out of 5840 were labeled “normal”

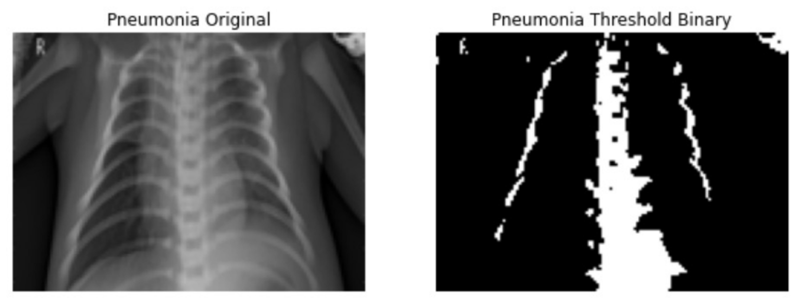
“Pneumonia” Verse “Normal” Images

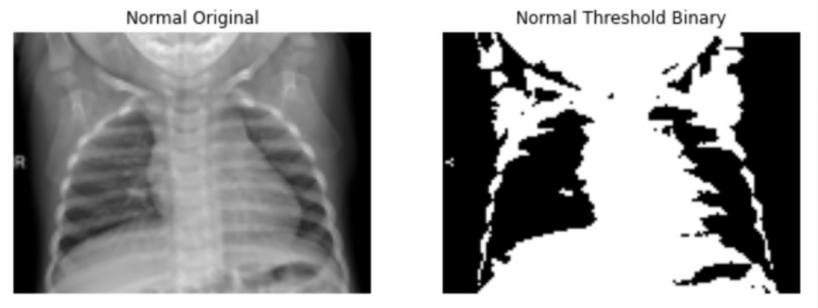


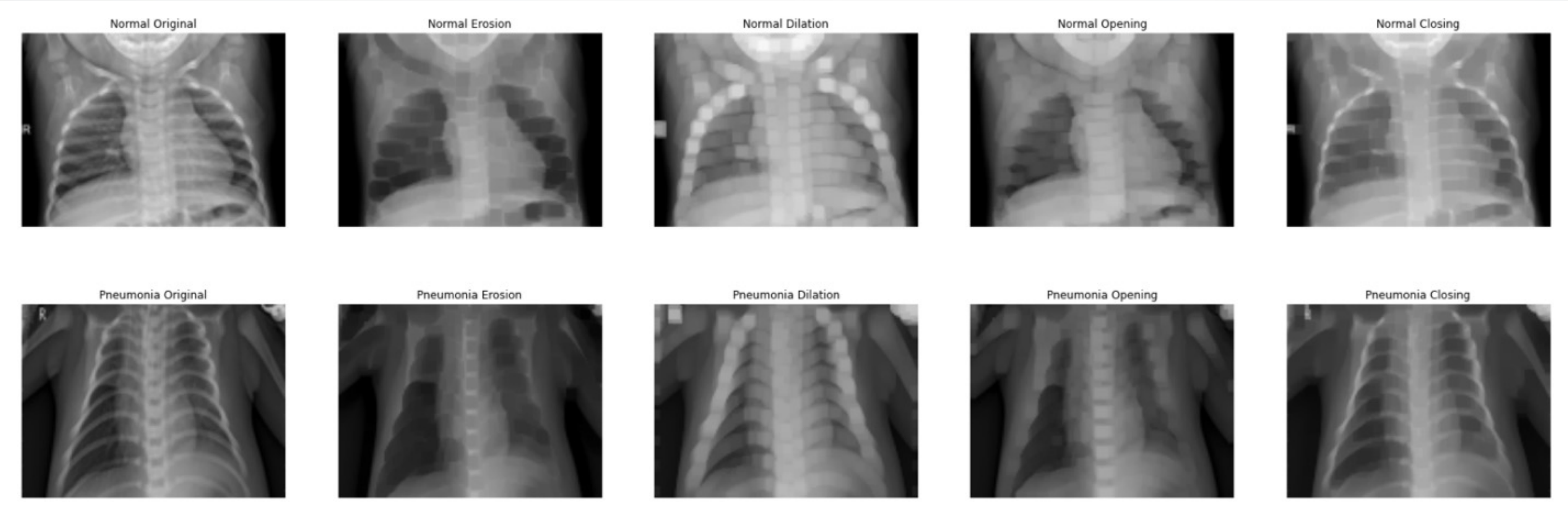
Exploratory Data Analysis Step 2











Advanced Analytics and Insights

Sklearn models with PCA

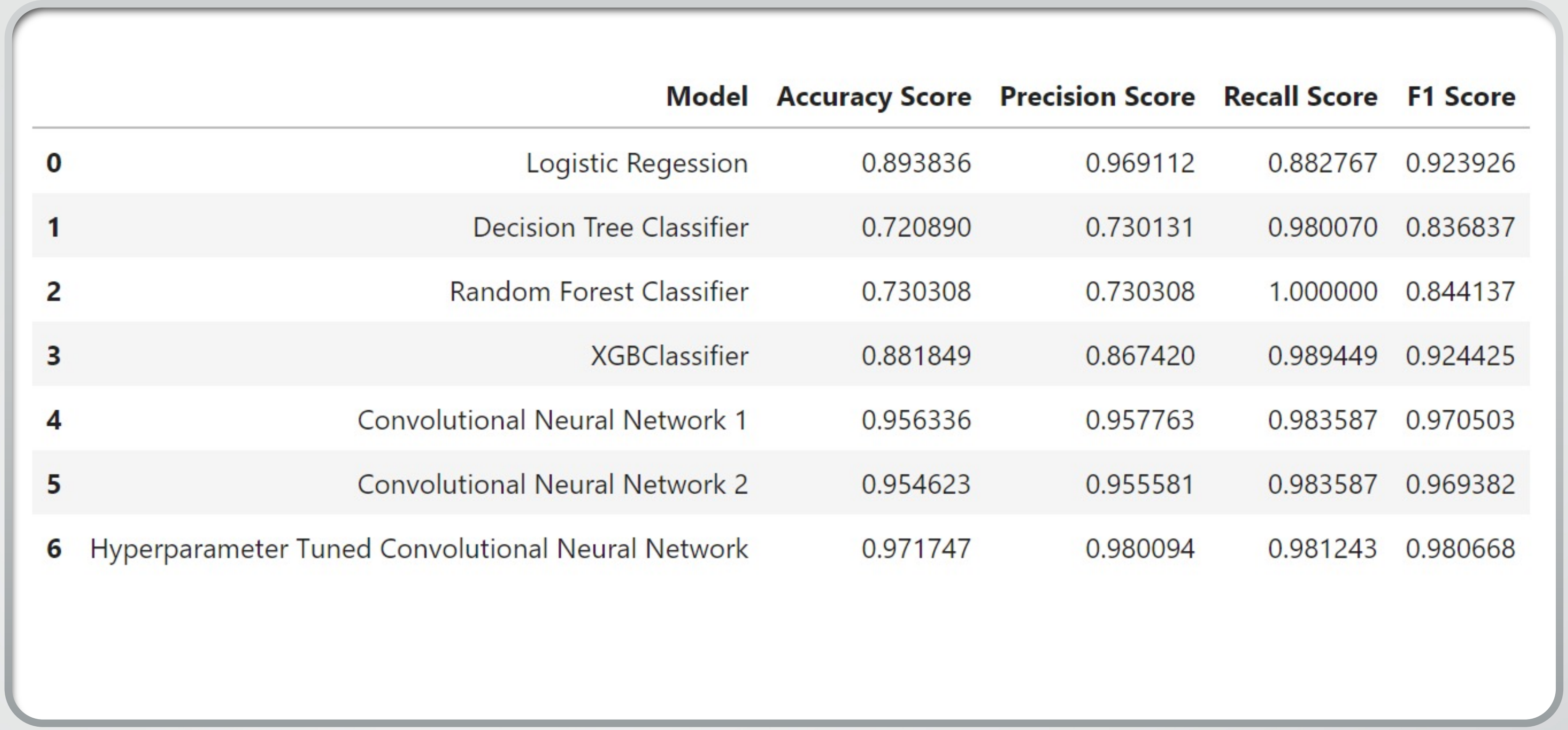
Linear Regression, Decision Tree, Random Forest, and XGBoost.

Tensor Flow to create Convolutional Neutral Networks

Two models that were not hyperparameter tuned

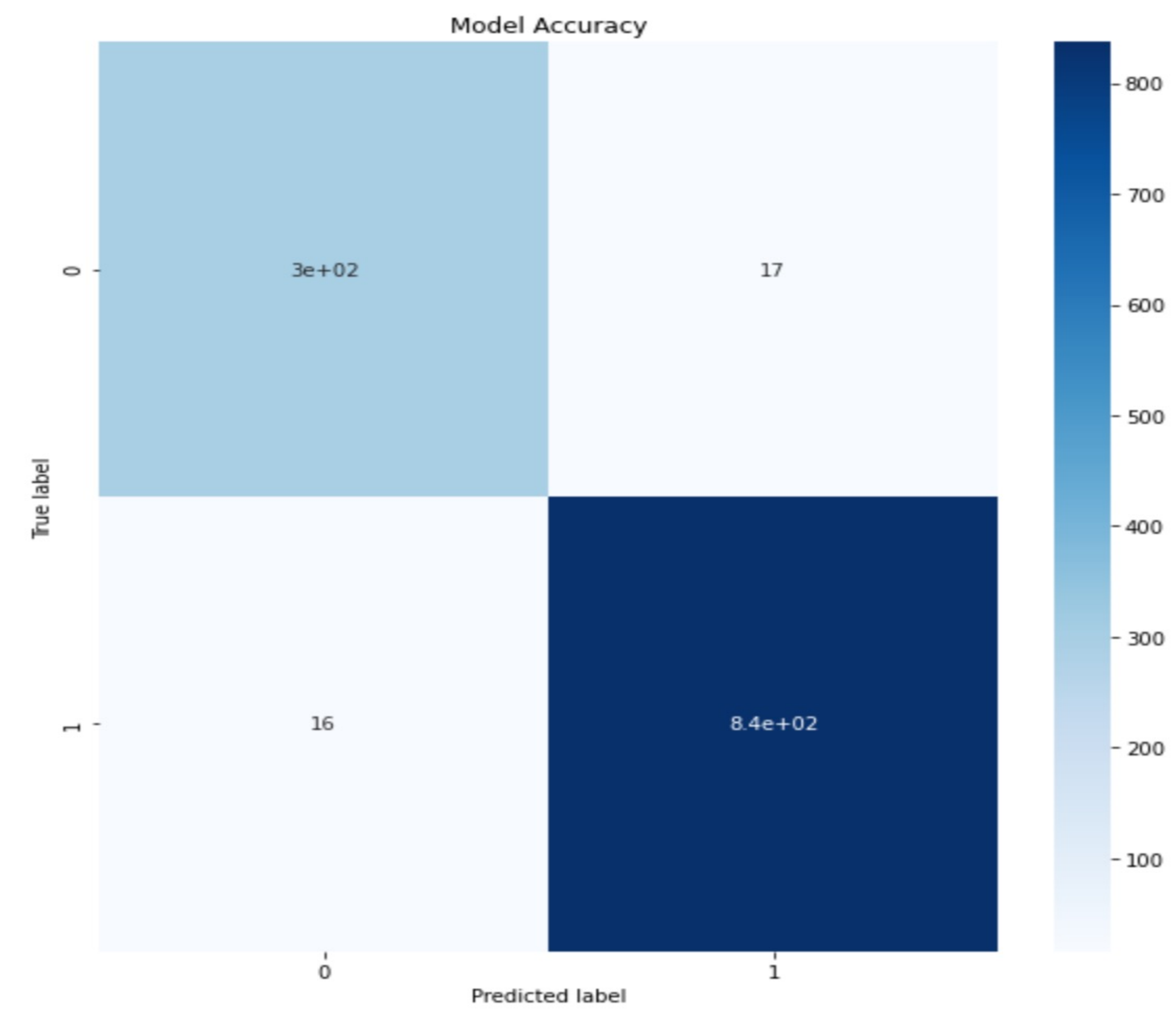
hyperparameter tuned Convolutional Neutral Network with Keras Tuner

Findings

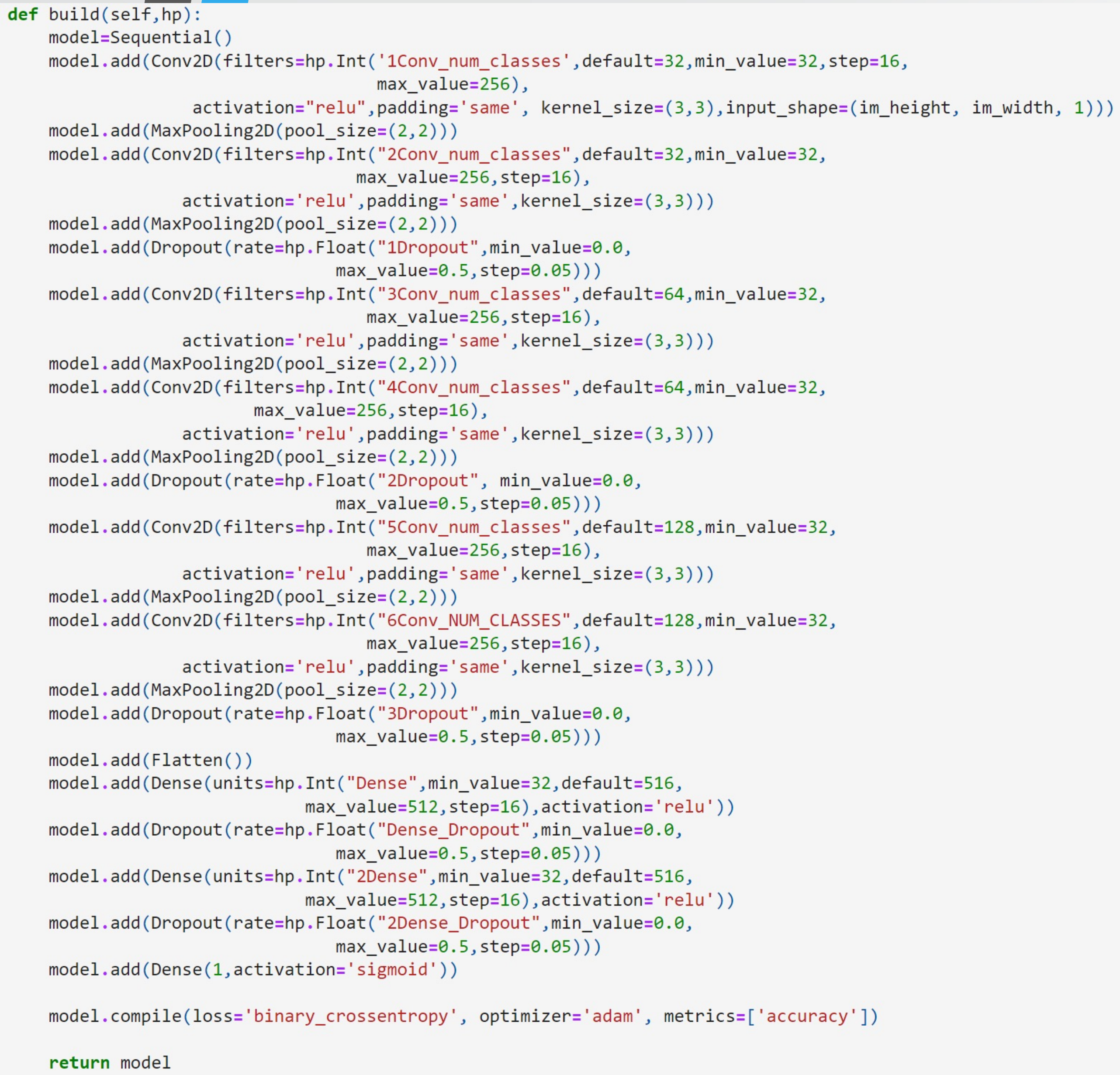


	Model	Accuracy Score	Precision Score	Recall Score	F1 Score
0	Logistic Regession	0.893836	0.969112	0.882767	0.923926
1	Decision Tree Classifier	0.720890	0.730131	0.980070	0.836837
2	Random Forest Classifier	0.730308	0.730308	1.000000	0.844137
3	XGBClassifier	0.881849	0.867420	0.989449	0.924425
4	Convolutional Neural Network 1	0.956336	0.957763	0.983587	0.970503
5	Convolutional Neural Network 2	0.954623	0.955581	0.983587	0.969382
6	Hyperparameter Tuned Convolutional Neural Network	0.971747	0.980094	0.981243	0.980668

Confusion Matrix



Final Model



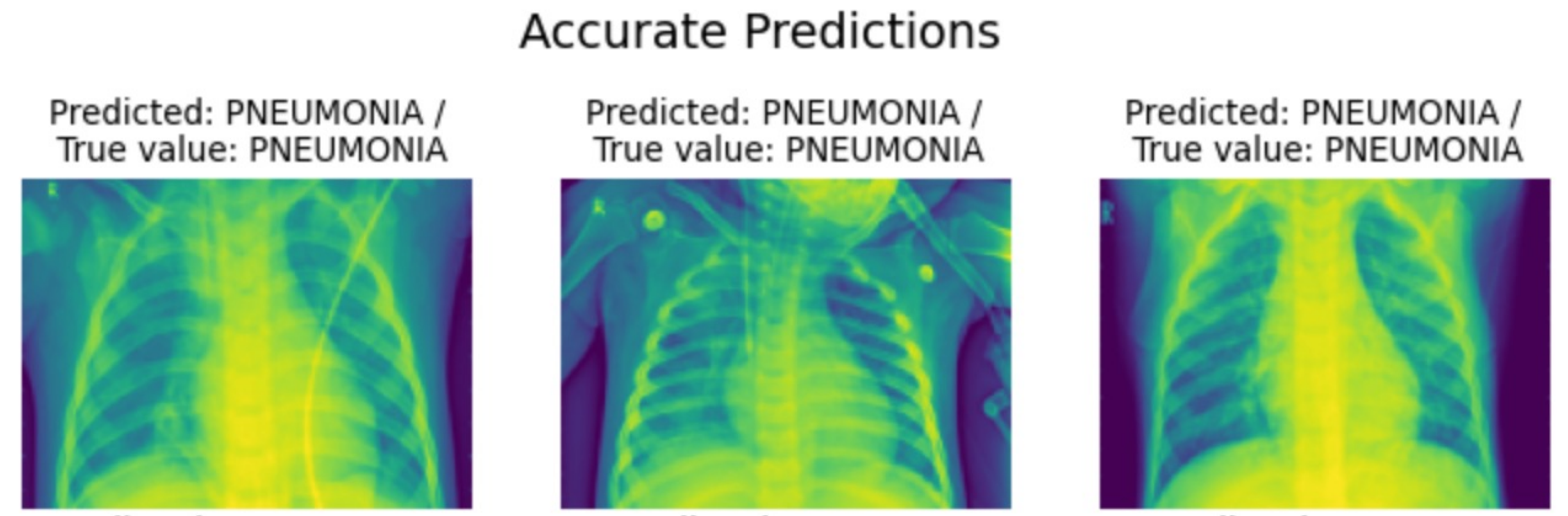
```python
def build(self,hp):
    model=Sequential()
    model.add(Conv2D(filters=hp.Int('1Conv_num_classes',default=32,min_value=32,step=16,
                                    max_value=256),
                  activation="relu",padding='same', kernel_size=(3,3),input_shape=(im_height, im_width, 1)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(filters=hp.Int("2Conv_num_classes",default=32,min_value=32,
                                  max_value=256,step=16),
                 activation='relu',padding='same',kernel_size=(3,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(rate=hp.Float("1Dropout",min_value=0.0,
                                max_value=0.5,step=0.05)))
    model.add(Conv2D(filters=hp.Int("3Conv_num_classes",default=64,min_value=32,
                                   max_value=256,step=16),
                 activation='relu',padding='same',kernel_size=(3,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(filters=hp.Int("4Conv_num_classes",default=64,min_value=32,
                        max_value=256,step=16),
                 activation='relu',padding='same',kernel_size=(3,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(rate=hp.Float("2Dropout", min_value=0.0,
                                max_value=0.5,step=0.05)))
    model.add(Conv2D(filters=hp.Int("5Conv_num_classes",default=128,min_value=32,
                                   max_value=256,step=16),
                 activation='relu',padding='same',kernel_size=(3,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(filters=hp.Int("6Conv_NUM_CLASSES",default=128,min_value=32,
                                   max_value=256,step=16),
                 activation='relu',padding='same',kernel_size=(3,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(rate=hp.Float("3Dropout",min_value=0.0,
                                max_value=0.5,step=0.05)))
    model.add(Flatten())
    model.add(Dense(units=hp.Int("Dense",min_value=32,default=516,
                             max_value=512,step=16),activation='relu'))
    model.add(Dropout(rate=hp.Float("Dense_Dropout",min_value=0.0,
                                max_value=0.5,step=0.05)))
    model.add(Dense(units=hp.Int("2Dense",min_value=32,default=516,
                             max_value=512,step=16),activation='relu'))
    model.add(Dropout(rate=hp.Float("2Dense_Dropout",min_value=0.0,
                                max_value=0.5,step=0.05)))
    model.add(Dense(1,activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model
```

```
Best Model Parameter:

"values": {"1Conv_num_classes": 96,
"2Conv_num_classes": 64, "1Dropout": 0.0,
"3Conv_num_classes": 144,
"4Conv_num_classes": 32, "2Dropout":
0.35000000000000003, "5Conv_num_classes":
144, "6Conv_NUM_CLASSES": 48, "3Dropout":
0.1, "Dense": 144, "Dense_Dropout": 0.1,
"2Dense": 160, "2Dense_Dropout": 0.05}
```

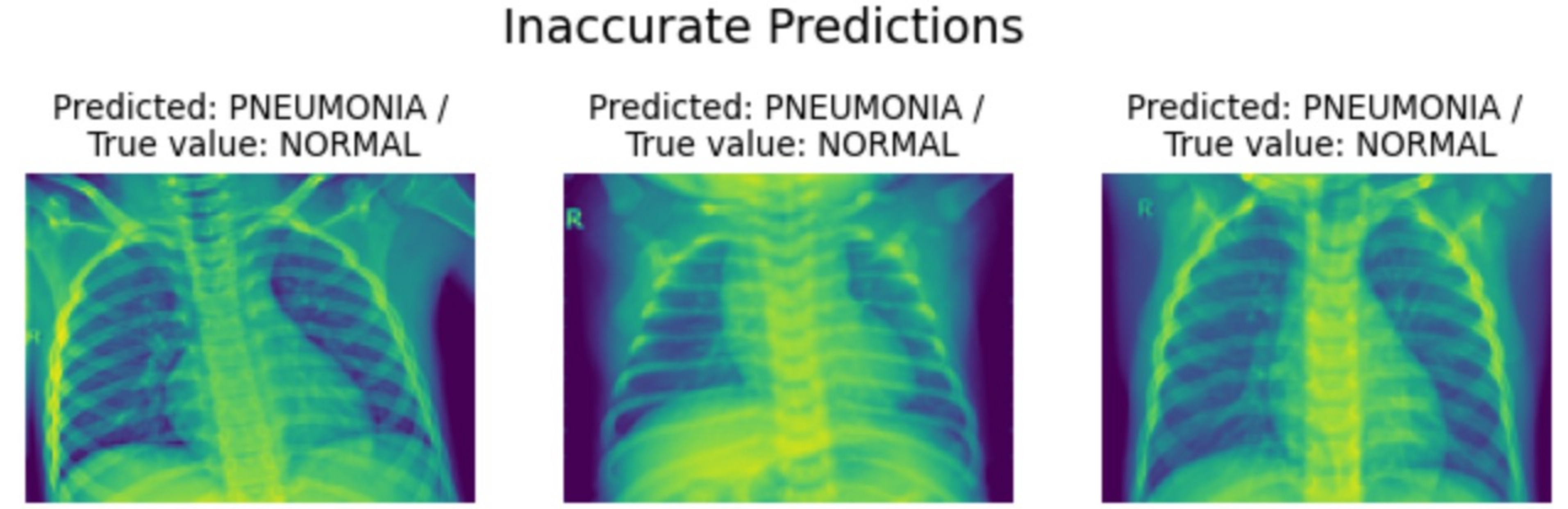


Accurate Predictions

Predicted: PNEUMONIA / True value: PNEUMONIA

Predicted: PNEUMONIA / True value: PNEUMONIA

Predicted: PNEUMONIA / True value: PNEUMONIA



Inaccurate Predictions

Predicted: PNEUMONIA / True value: NORMAL

Predicted: PNEUMONIA / True value: NORMAL

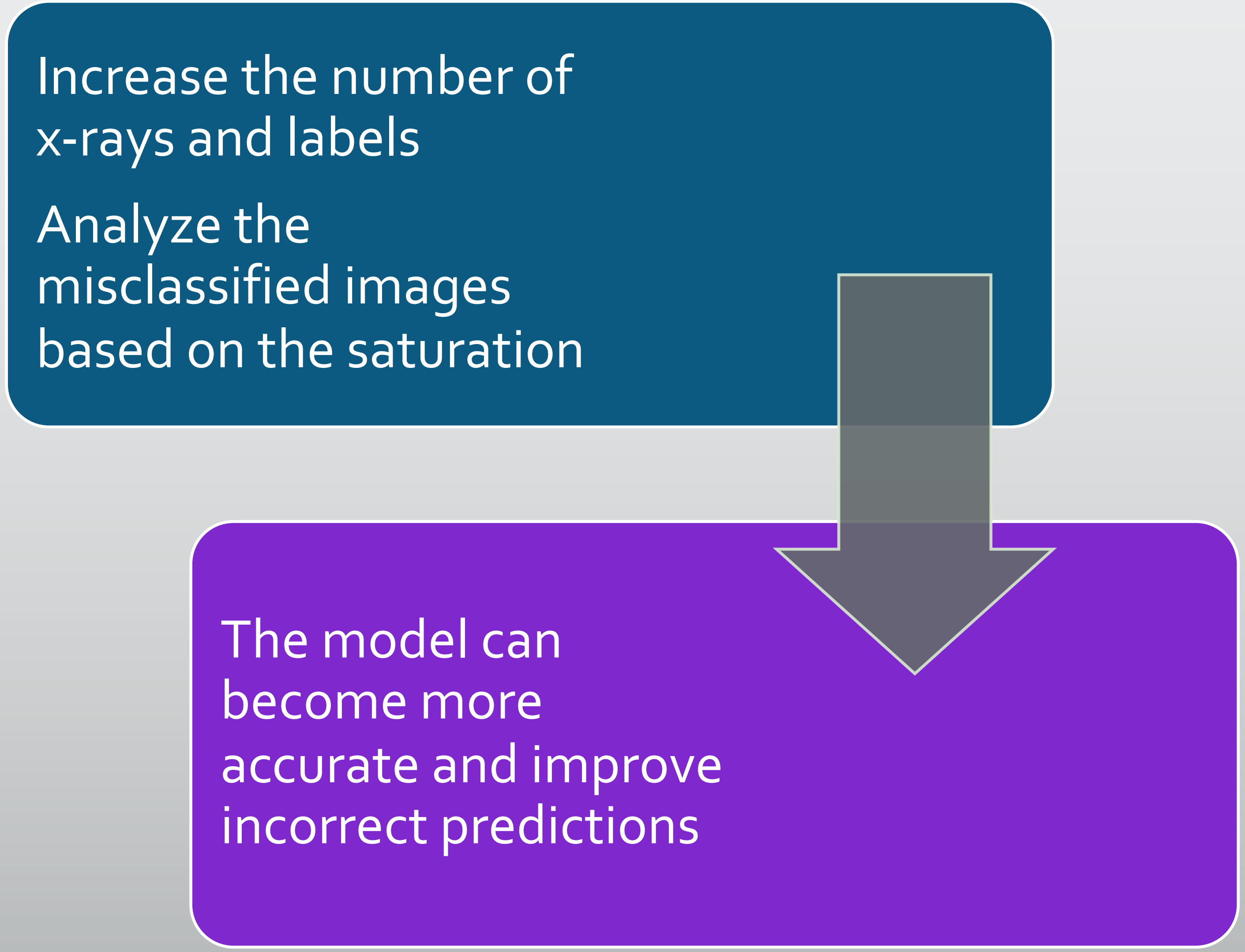
Predicted: PNEUMONIA / True value: NORMAL

Recommendations

Predict the presents of pneumonia in a patient’s chest x-ray with 97% accuracy

Will lead to few misclassification

Future work



Increase the number of x-rays and labels

Analyze the misclassified images based on the saturation

The model can become more accurate and improve incorrect predictions