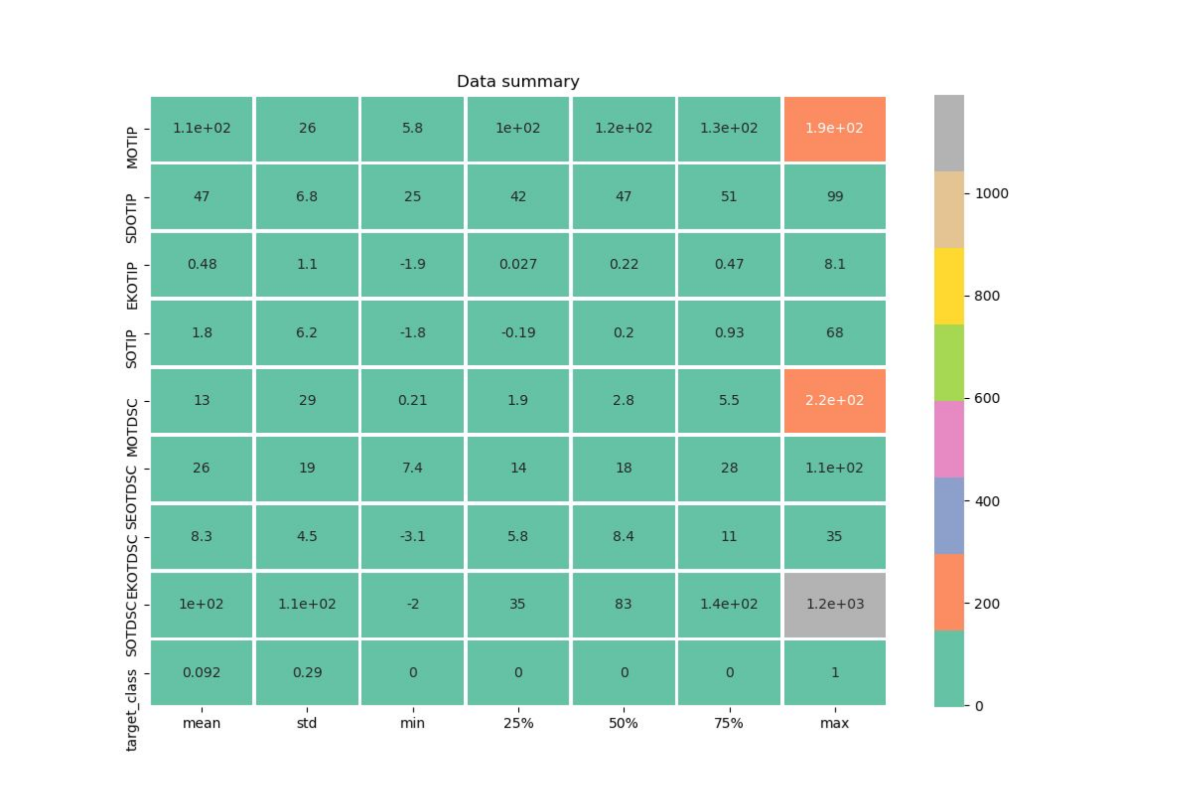
Pulsars prediction

1. Checking the dataset

We will need to scale it.

Using **StandardScaler**

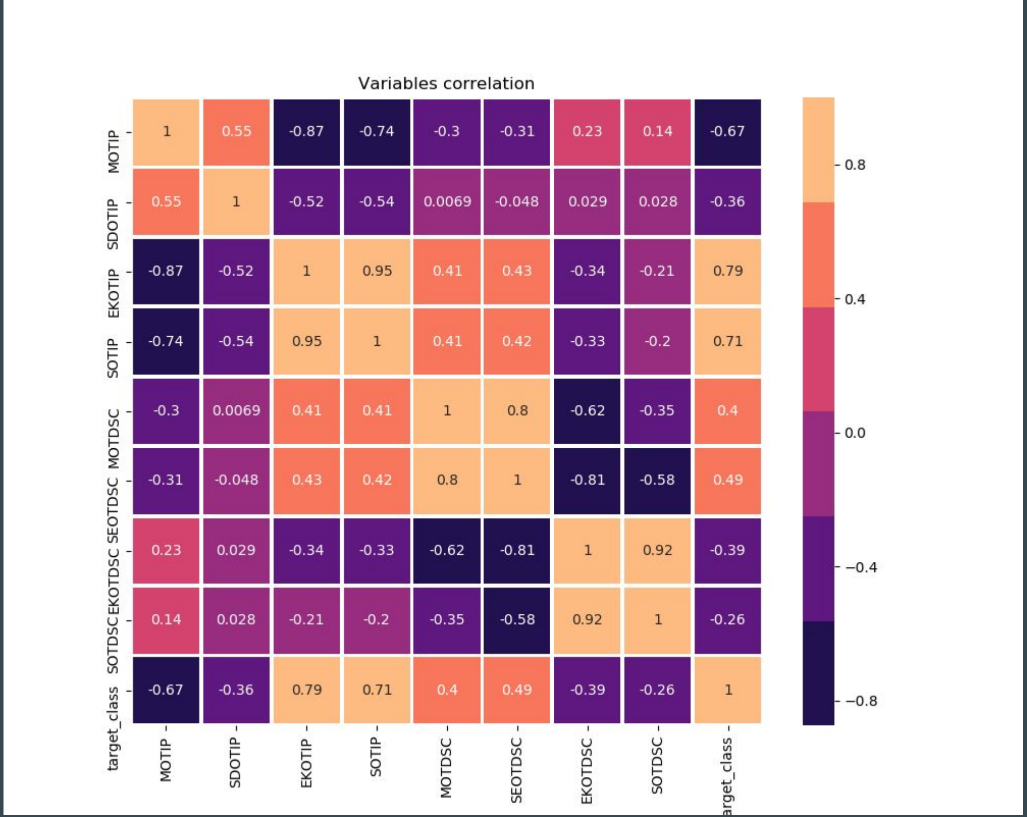


Data summary

	mean	std	min	25%	50%	75%	max
MOTIP	1.1e+02	26	5.8	1e+02	1.2e+02	1.3e+02	1.9e+02
SDOTIP	47	6.8	25	42	47	51	99
EKOTIP	0.48	1.1	-1.9	0.027	0.22	0.47	8.1
SOTIP	1.8	6.2	-1.8	-0.19	0.2	0.93	68
MOTDSC	13	29	0.21	1.9	2.8	5.5	2.2e+02
SEOTDSC	26	19	7.4	14	18	28	1.1e+02
EKOTDSC	8.3	4.5	-3.1	5.8	8.4	11	35
SOTDSC	1e+02	1.1e+02	-2	35	83	1.4e+02	1.2e+03
target_class	0.092	0.29	0	0	0	0	1

1. Checking the dataset

There is a strong correlation between **SOTIP** and **EKOTIP** and also between **SOTDSC** and **EKOTDSC**, so we will drop **SOTIP** and **SOTDSC**

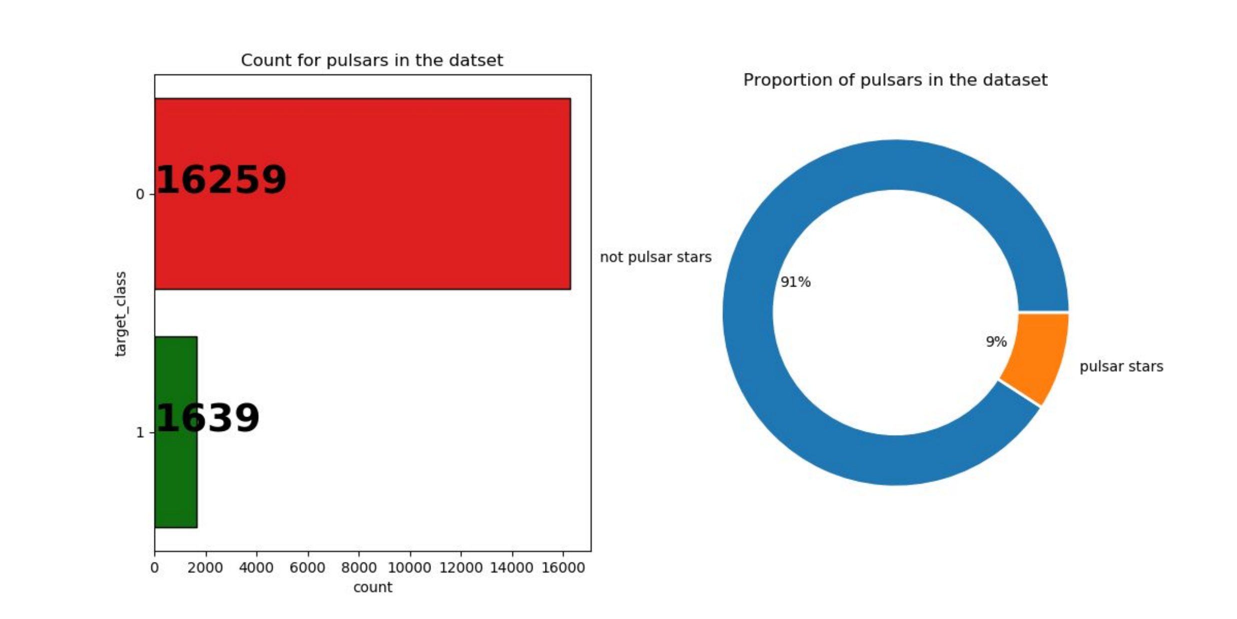


Variables correlation

	MOTIP	SDOTIP	EKOTIP	SOTIP	MOTDSC	SEOTDSC	EKOTDSC	SOTDSC	arget_class
MOTIP	1	0.55	-0.87	-0.74	-0.3	-0.31	0.23	0.14	-0.67
SDOTIP	0.55	1	-0.52	-0.54	0.0069	-0.048	0.029	0.028	-0.36
EKOTIP	-0.87	-0.52	1	0.95	0.41	0.43	-0.34	-0.21	0.79
SOTIP	-0.74	-0.54	0.95	1	0.41	0.42	-0.33	-0.2	0.71
MOTDSC	-0.3	0.0069	0.41	0.41	1	0.8	-0.62	-0.35	0.4
SEOTDSC	-0.31	-0.048	0.43	0.42	0.8	1	-0.81	-0.58	0.49
EKOTDSC	0.23	0.029	-0.34	-0.33	-0.62	-0.81	1	0.92	-0.39
SOTDSC	0.14	0.028	-0.21	-0.2	-0.35	-0.58	0.92	1	-0.26
target_class	-0.67	-0.36	0.79	0.71	0.4	0.49	-0.39	-0.26	1

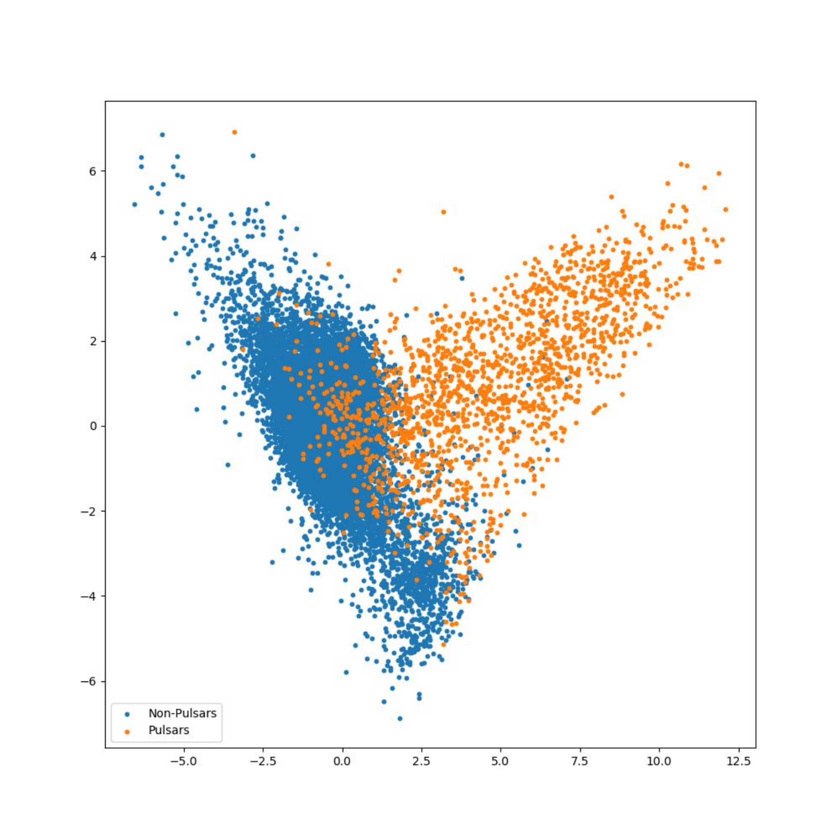
1. Checking the dataset

Dataset is imbalanced - we will need to use stratification



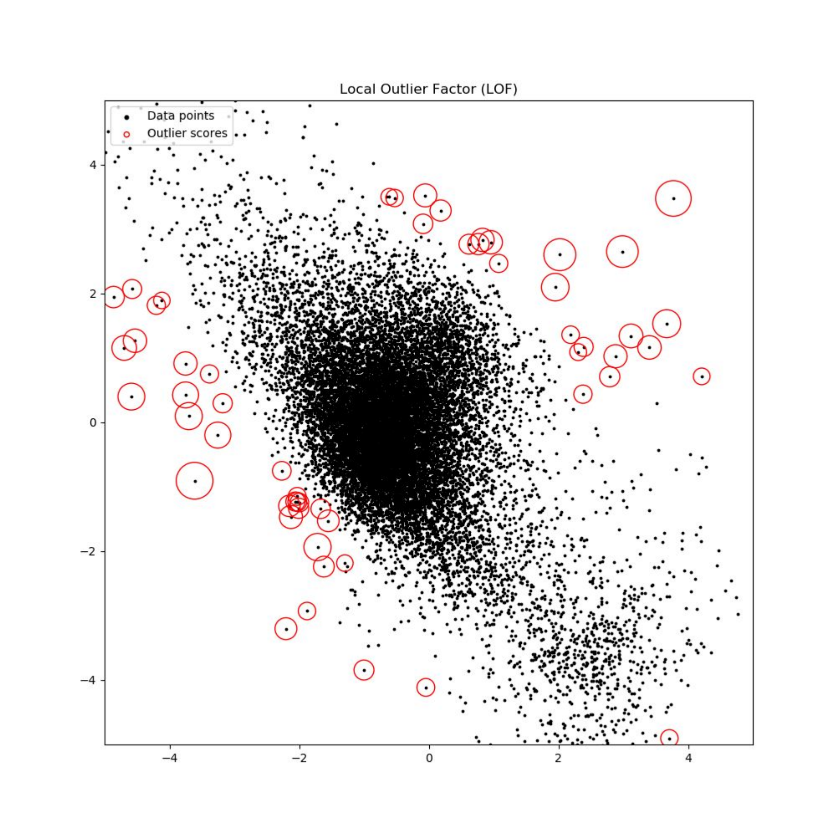
1. Checking the dataset

Let's search for anomalies using Principal Component Analysis and Local Outlier Factor



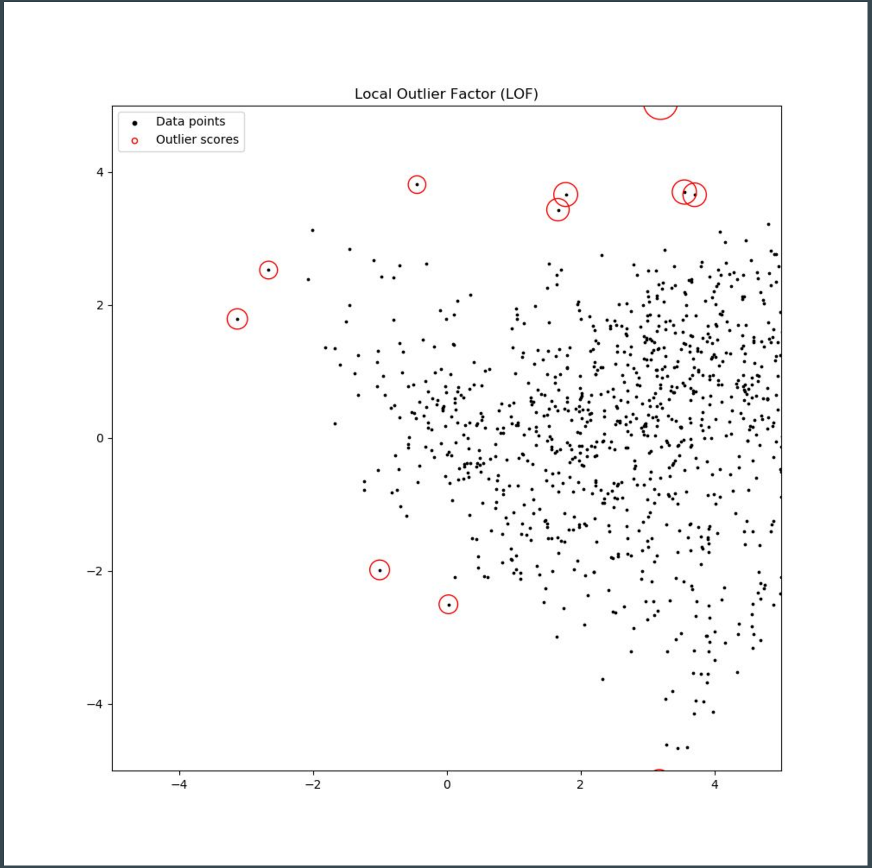
1. Checking the dataset

We can see some outliers for non-pulsars...



1. Checking the dataset

...and for pulsars too.
Let's remove them.



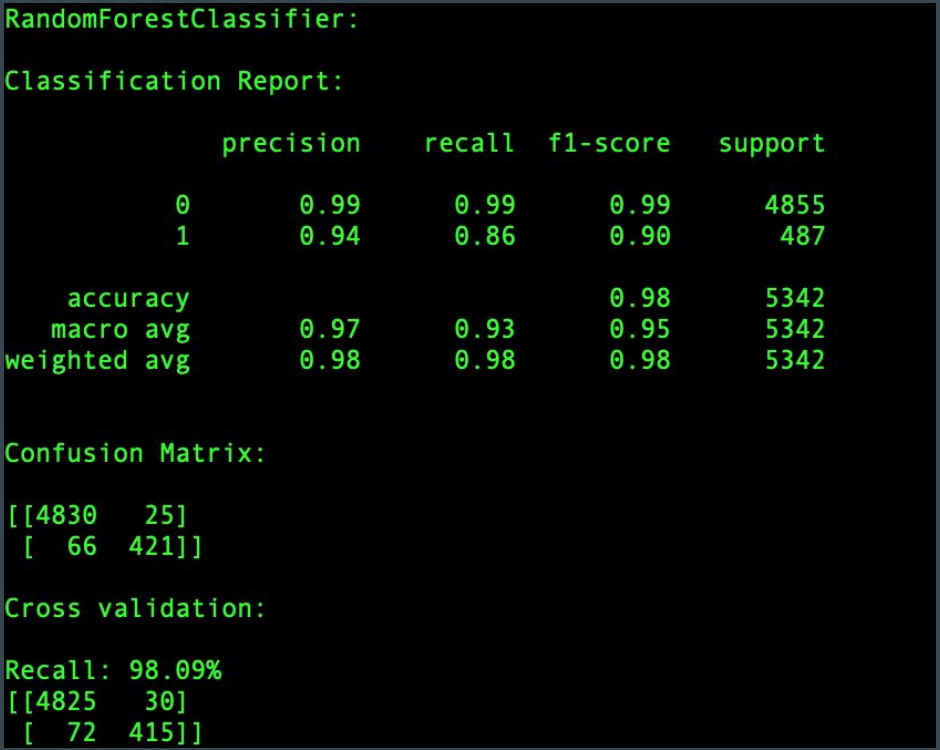
1. Checking the dataset

Splitting the dataset

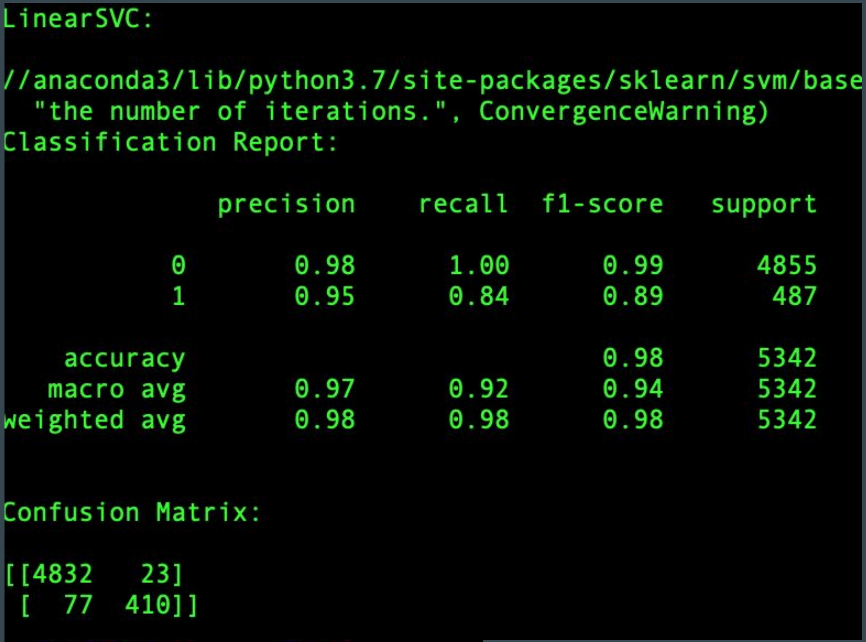
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y,
random_state=0)
```

And then go on testing different models

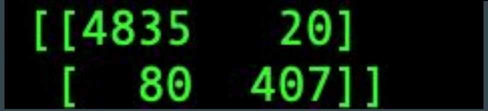
2. Testing models



```
RandomForestClassifier:

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4855
           1       0.94      0.86      0.90       487

    accuracy                           0.98      5342
   macro avg       0.97      0.93      0.95      5342
weighted avg       0.98      0.98      0.98      5342


Confusion Matrix:

[[4830   25]
 [  66  421]]

Cross validation:

Recall: 98.09%
[[4825   30]
 [  72  415]]
```

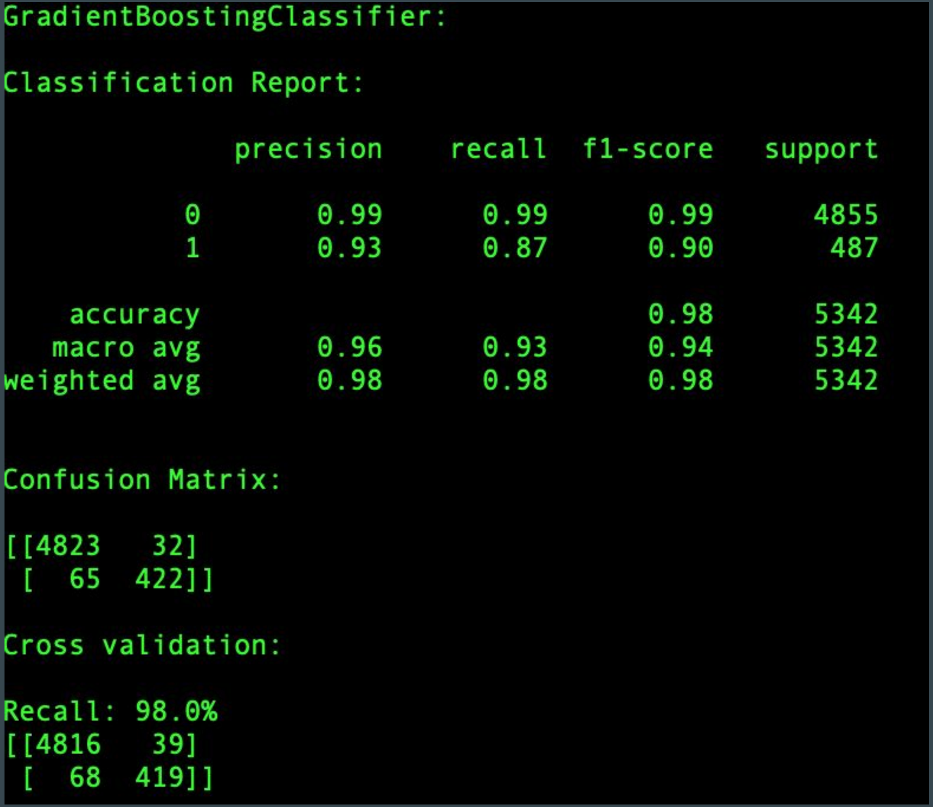


```
LinearSVC:

//anaconda3/lib/python3.7/site-packages/sklearn/svm/base
  "the number of iterations.", ConvergenceWarning)
Classification Report:

              precision    recall  f1-score   support

           0       0.98      1.00      0.99      4855
           1       0.95      0.84      0.89       487

    accuracy                           0.98      5342
   macro avg       0.97      0.92      0.94      5342
weighted avg       0.98      0.98      0.98      5342


Confusion Matrix:

[[4832   23]
 [  77  410]]
```



```
[[4835   20]
 [  80  407]]
```

2. Testing models



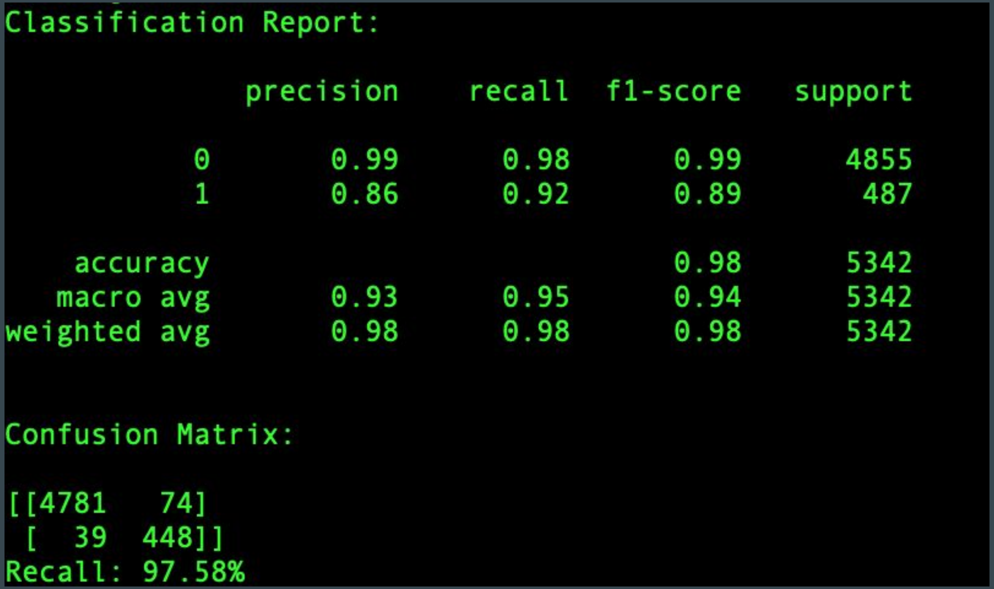
```
GradientBoostingClassifier:

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4855
           1       0.93      0.87      0.90       487

    accuracy                           0.98      5342
   macro avg       0.96      0.93      0.94      5342
weighted avg       0.98      0.98      0.98      5342


Confusion Matrix:

[[4823   32]
 [  65  422]]

Cross validation:

Recall: 98.0%
[[4816   39]
 [  68  419]]
```

3. Tuning RF using **GridSearchCV**

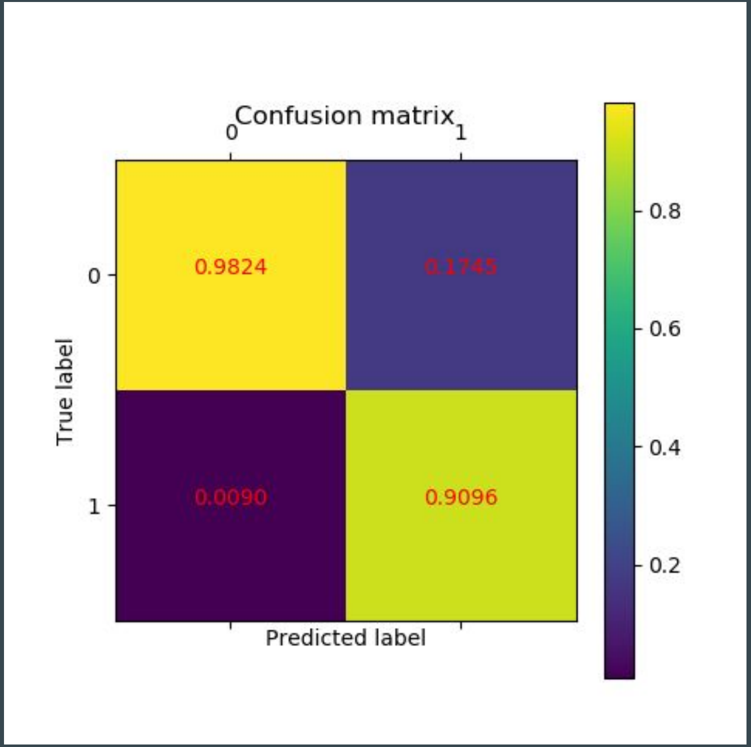
```
forest = RandomForestClassifier(bootstrap=True, class_weight='balanced_subsample',
                      criterion='gini', max_depth=15, max_features=4,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=40, min_weight_fraction_leaf=0.0,
                      n_estimators=120, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

3. Tuning RF using **GridSearchCV**

Tuning to make FN minimal

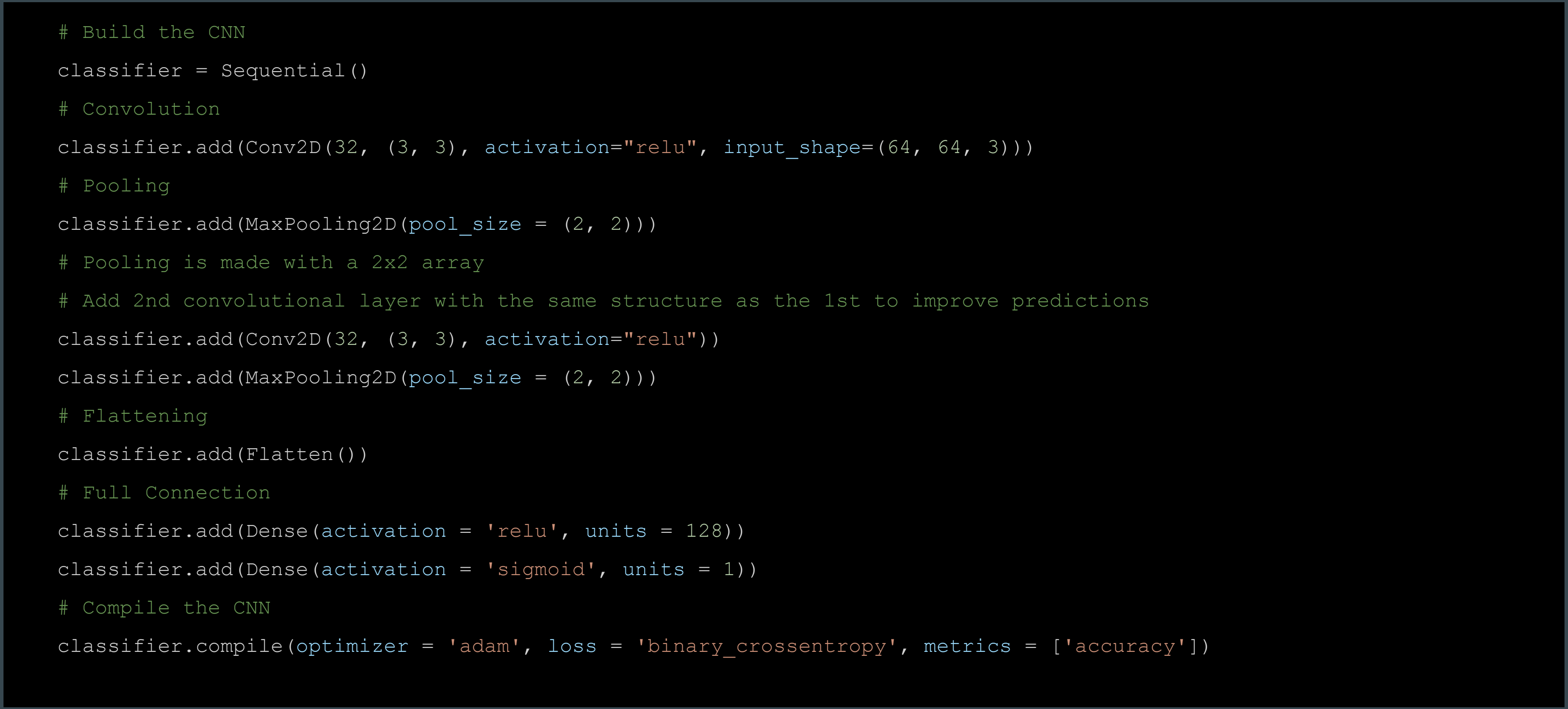


```
Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4855
           1       0.86      0.92      0.89       487

    accuracy                           0.98      5342
   macro avg       0.93      0.95      0.94      5342
weighted avg       0.98      0.98      0.98      5342


Confusion Matrix:

[[4781   74]
 [  39  448]]
Recall: 97.58%
```



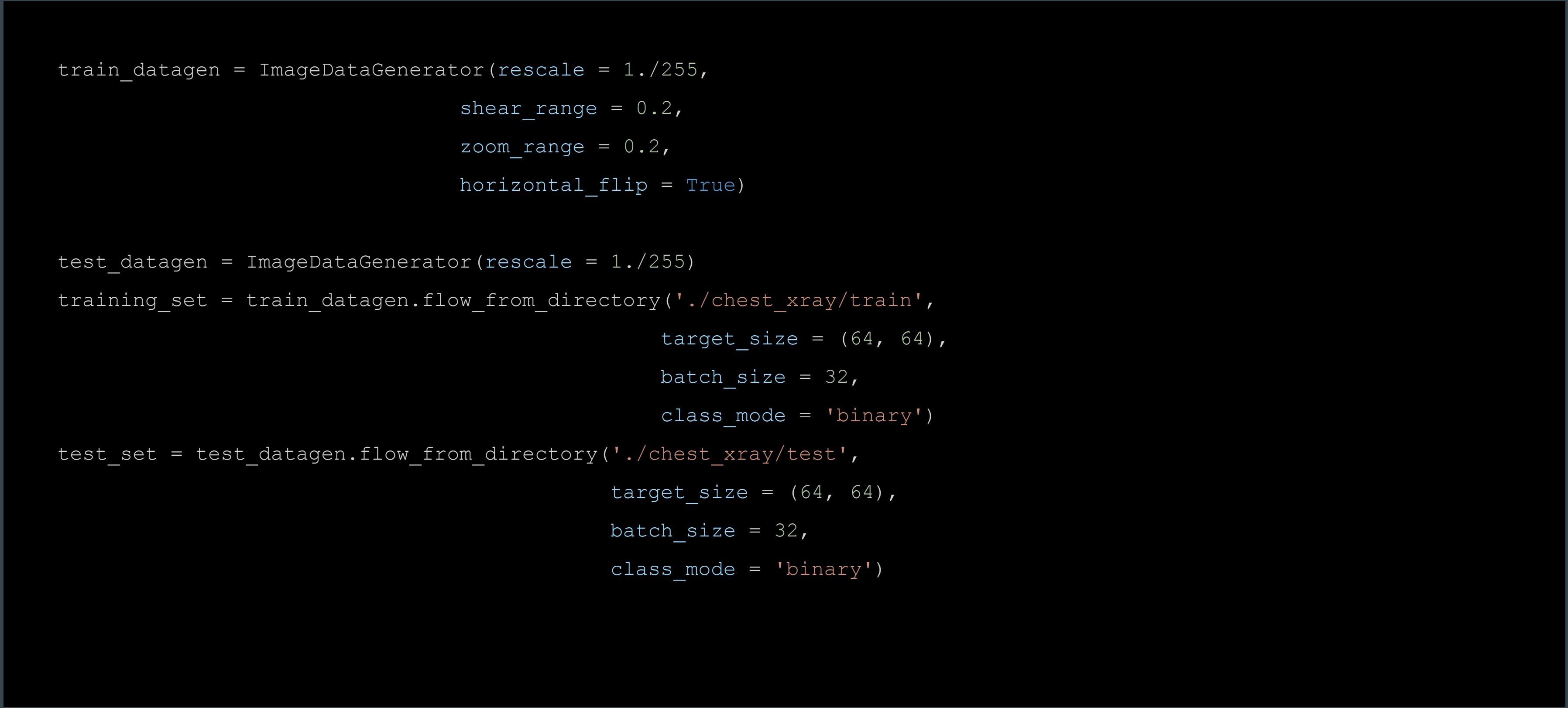
Pneumonia prediction

Using CNN



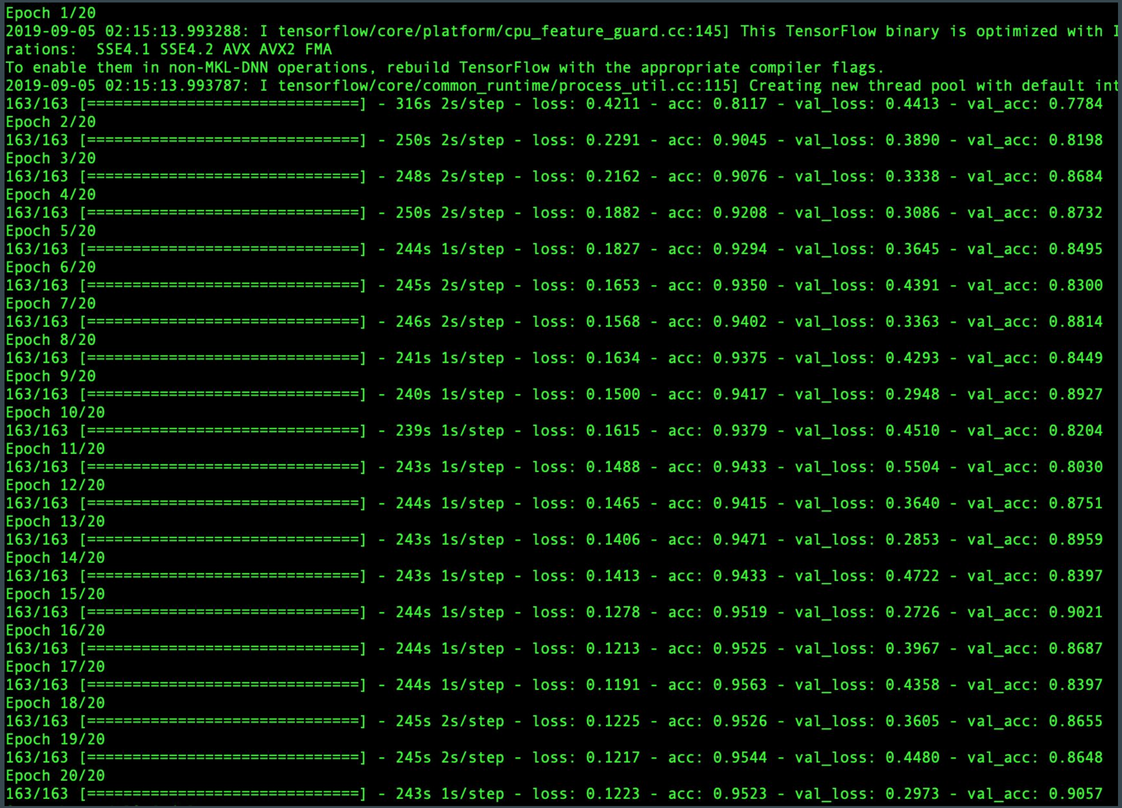
```
# Build the CNN
classifier = Sequential()
# Convolution
classifier.add(Conv2D(32, (3, 3), activation="relu", input_shape=(64, 64, 3)))
# Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Pooling is made with a 2x2 array
# Add 2nd convolutional layer with the same structure as the 1st to improve predictions
classifier.add(Conv2D(32, (3, 3), activation="relu"))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Flattening
classifier.add(Flatten())
# Full Connection
classifier.add(Dense(activation = 'relu', units = 128))
classifier.add(Dense(activation = 'sigmoid', units = 1))
# Compile the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Preparing Image Generator



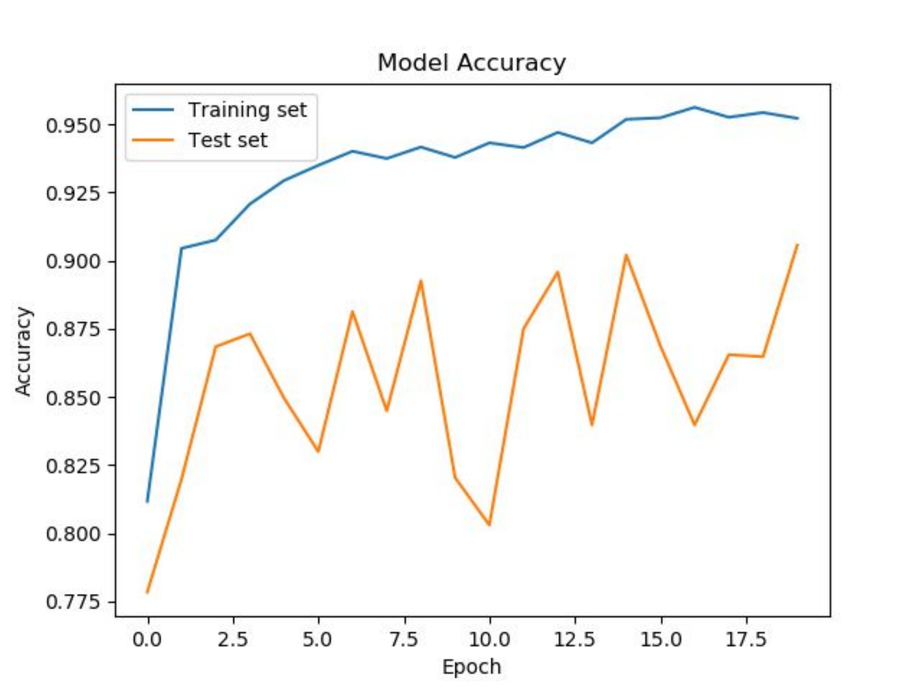
```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                shear_range = 0.2,
                                zoom_range = 0.2,
                                horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
training_set = train_datagen.flow_from_directory('./chest_xray/train',
                                                target_size = (64, 64),
                                                batch_size = 32,
                                                class_mode = 'binary')
test_set = test_datagen.flow_from_directory('./chest_xray/test',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

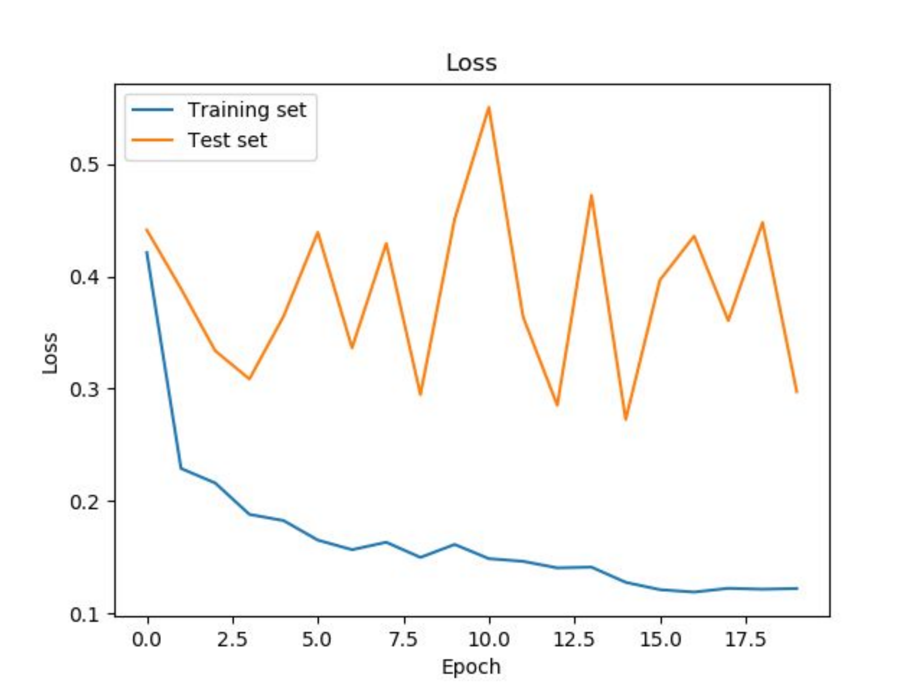
Running on 20 epoch



```
Epoch 1/20
2019-09-05 02:15:13.993288: I tensorflow/core/platform/cpu_feature_guard.cc:145] This TensorFlow binary is optimized with I
rations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2019-09-05 02:15:13.993787: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with default int
163/163 [==============================] - 316s 2s/step - loss: 0.4211 - acc: 0.8117 - val_loss: 0.4413 - val_acc: 0.7784
Epoch 2/20
163/163 [==============================] - 250s 2s/step - loss: 0.2291 - acc: 0.9045 - val_loss: 0.3890 - val_acc: 0.8198
Epoch 3/20
163/163 [==============================] - 248s 2s/step - loss: 0.2162 - acc: 0.9076 - val_loss: 0.3338 - val_acc: 0.8684
Epoch 4/20
163/163 [==============================] - 250s 2s/step - loss: 0.1882 - acc: 0.9208 - val_loss: 0.3086 - val_acc: 0.8732
Epoch 5/20
163/163 [==============================] - 244s 1s/step - loss: 0.1827 - acc: 0.9294 - val_loss: 0.3645 - val_acc: 0.8495
Epoch 6/20
163/163 [==============================] - 245s 2s/step - loss: 0.1653 - acc: 0.9350 - val_loss: 0.4391 - val_acc: 0.8300
Epoch 7/20
163/163 [==============================] - 246s 2s/step - loss: 0.1568 - acc: 0.9402 - val_loss: 0.3363 - val_acc: 0.8814
Epoch 8/20
163/163 [==============================] - 241s 1s/step - loss: 0.1634 - acc: 0.9375 - val_loss: 0.4293 - val_acc: 0.8449
Epoch 9/20
163/163 [==============================] - 240s 1s/step - loss: 0.1500 - acc: 0.9417 - val_loss: 0.2948 - val_acc: 0.8927
Epoch 10/20
163/163 [==============================] - 239s 1s/step - loss: 0.1615 - acc: 0.9379 - val_loss: 0.4510 - val_acc: 0.8204
Epoch 11/20
163/163 [==============================] - 243s 1s/step - loss: 0.1488 - acc: 0.9433 - val_loss: 0.5504 - val_acc: 0.8030
Epoch 12/20
163/163 [==============================] - 244s 1s/step - loss: 0.1465 - acc: 0.9415 - val_loss: 0.3640 - val_acc: 0.8751
Epoch 13/20
163/163 [==============================] - 243s 1s/step - loss: 0.1406 - acc: 0.9471 - val_loss: 0.2853 - val_acc: 0.8959
Epoch 14/20
163/163 [==============================] - 243s 1s/step - loss: 0.1413 - acc: 0.9433 - val_loss: 0.4722 - val_acc: 0.8397
Epoch 15/20
163/163 [==============================] - 244s 1s/step - loss: 0.1278 - acc: 0.9519 - val_loss: 0.2726 - val_acc: 0.9021
Epoch 16/20
163/163 [==============================] - 244s 1s/step - loss: 0.1213 - acc: 0.9525 - val_loss: 0.3967 - val_acc: 0.8687
Epoch 17/20
163/163 [==============================] - 244s 1s/step - loss: 0.1191 - acc: 0.9563 - val_loss: 0.4358 - val_acc: 0.8397
Epoch 18/20
163/163 [==============================] - 245s 2s/step - loss: 0.1225 - acc: 0.9526 - val_loss: 0.3605 - val_acc: 0.8655
Epoch 19/20
163/163 [==============================] - 245s 2s/step - loss: 0.1217 - acc: 0.9544 - val_loss: 0.4480 - val_acc: 0.8648
Epoch 20/20
163/163 [==============================] - 243s 1s/step - loss: 0.1223 - acc: 0.9523 - val_loss: 0.2973 - val_acc: 0.9057
```

We can even continue training, accuracy growing


Model Accuracy
Training set
Test set
Accuracy
Epoch


Loss
Training set
Test set
Loss
Epoch