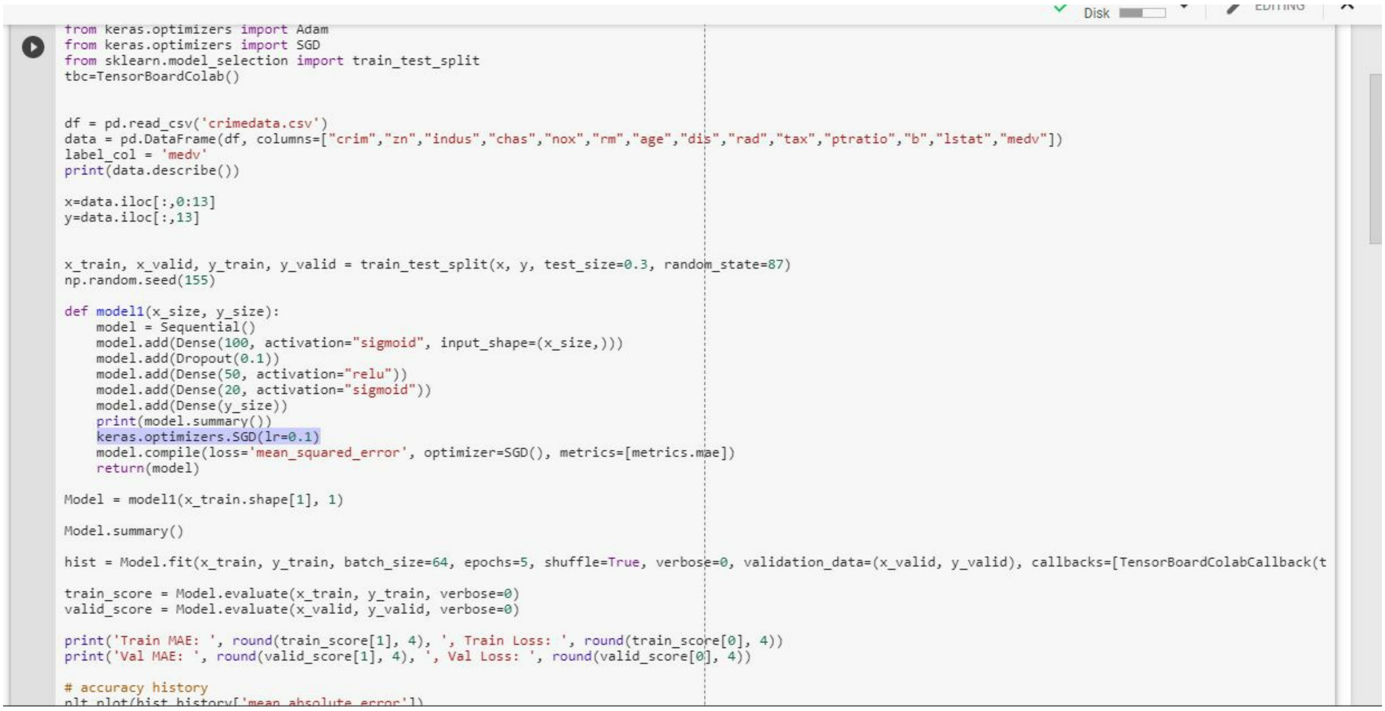
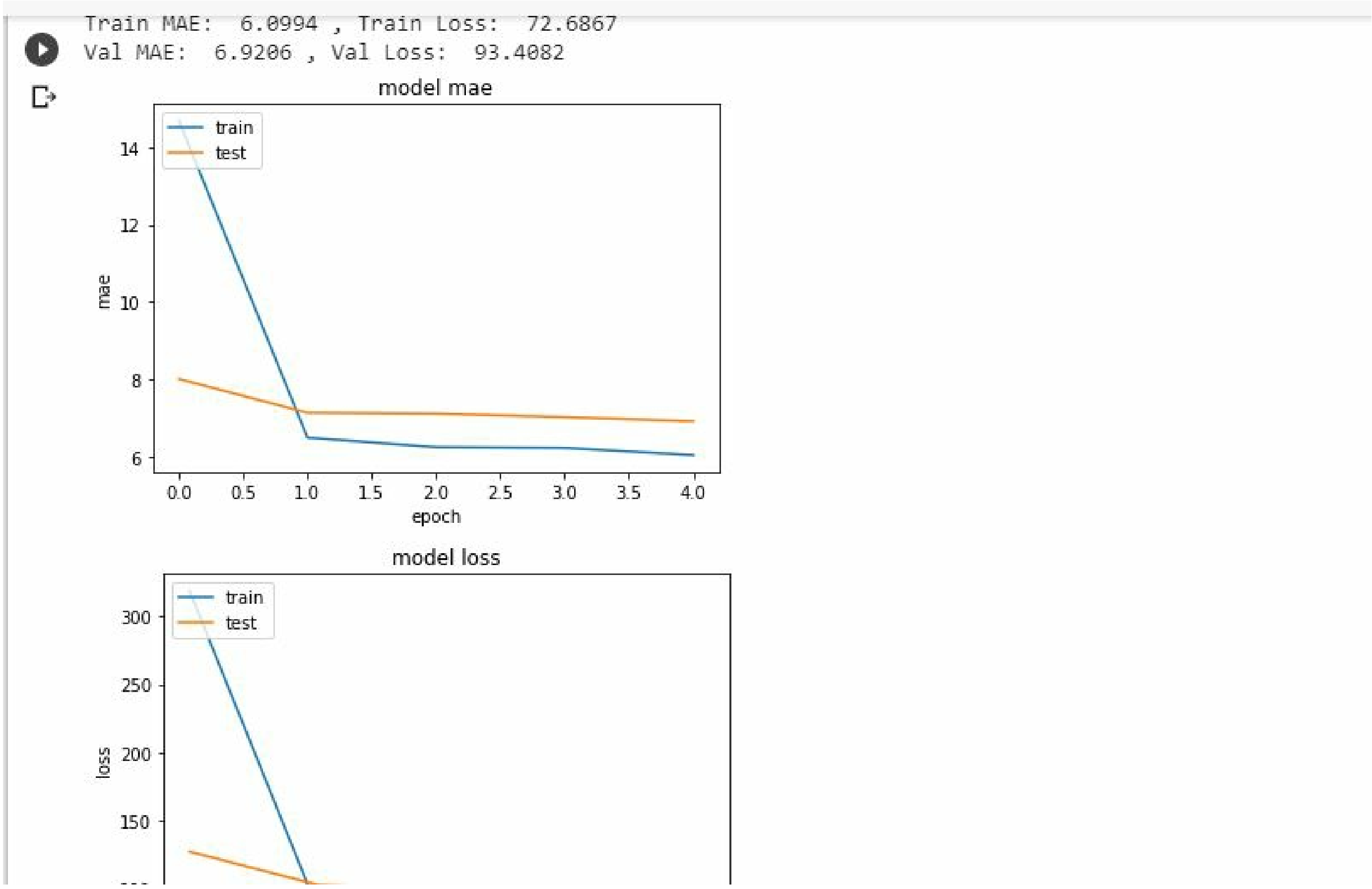
Team2

1. We have taken Crimedataset where ‘medv’ column is the target column depends on other features in the data. In the first instant We have used SGD optimizer,learning rate = 0.1,batch size = 64 and sigmoid activation function. In the second instant We have used Adam optimizer,learning rate = 0.2,batch size = 32 and tanh activation function. The loss has increased. It may be one of the factor that learning rate has increased so that model tries to train fast and accuracy decreases. The other factors are change in activation function, batch size and optimizer.

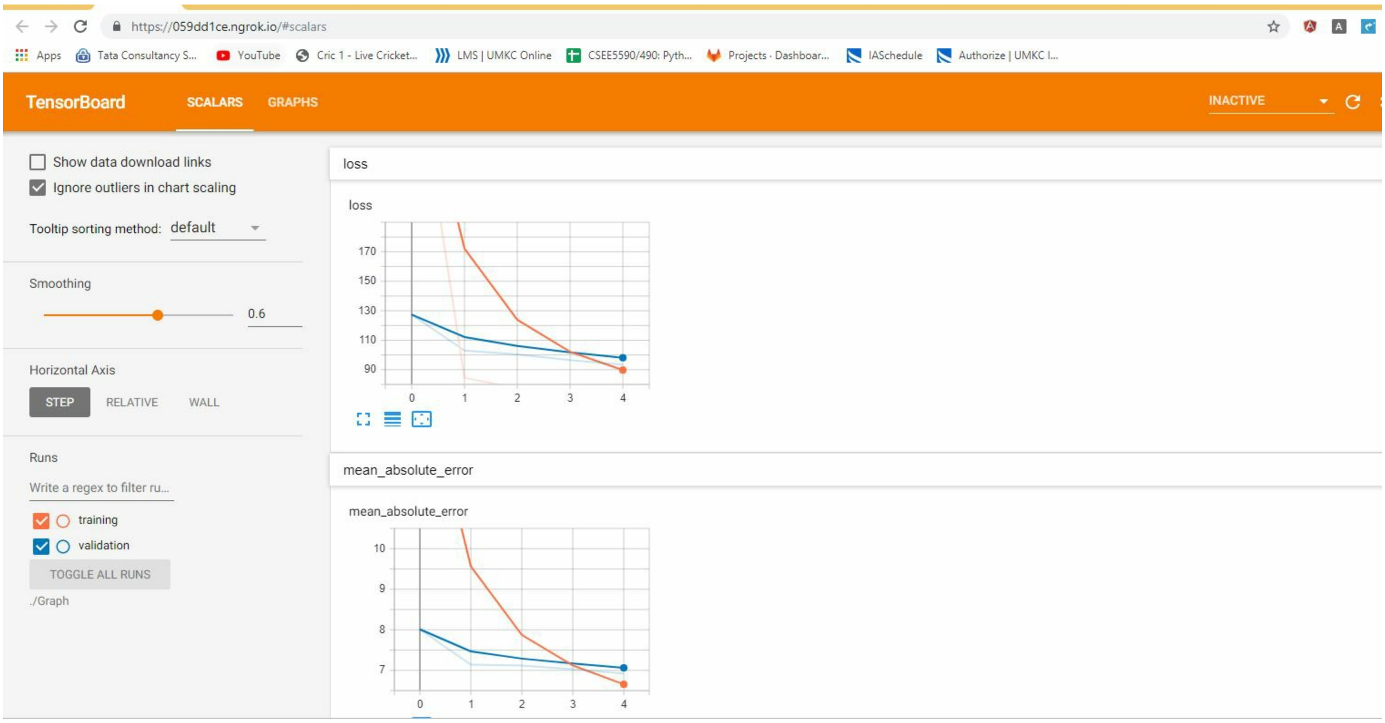
Please find the screenshots below



```
from keras.optimizers import Adam
from keras.optimizers import SGD
from sklearn.model_selection import train_test_split
tbc=TensorBoardColab()


df = pd.read_csv('crimedata.csv')
data = pd.DataFrame(df, columns=["crim","zn","indus","chas","nox","rm","age","dis","rad","tax","ptratio","b","lstat","medv"])
label_col = 'medv'
print(data.describe())

x=data.iloc[:,0:13]
y=data.iloc[:,13]


x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size=0.3, random_state=87)
np.random.seed(155)

def model1(x_size, y_size):
    model = Sequential()
    model.add(Dense(100, activation="sigmoid", input_shape=(x_size,)))
    model.add(Dropout(0.1))
    model.add(Dense(50, activation="relu"))
    model.add(Dense(20, activation="sigmoid"))
    model.add(Dense(y_size))
    print(model.summary())
    keras.optimizers.SGD(lr=0.1)
    model.compile(loss='mean_squared_error', optimizer=SGD(), metrics=[metrics.mae])
    return(model)

Model = model1(x_train.shape[1], 1)

Model.summary()

hist = Model.fit(x_train, y_train, batch_size=64, epochs=5, shuffle=True, verbose=0, validation_data=(x_valid, y_valid), callbacks=[TensorBoardColabCallback(t

train_score = Model.evaluate(x_train, y_train, verbose=0)
valid_score = Model.evaluate(x_valid, y_valid, verbose=0)

print('Train MAE: ', round(train_score[1], 4), ', Train Loss: ', round(train_score[0], 4))
print('Val MAE: ', round(valid_score[1], 4), ', Val Loss: ', round(valid_score[0], 4))

# accuracy history
plt.plot(hist.history['mean_absolute_error'])
```



```
Train MAE:  6.0994 , Train Loss:  72.6867
Val MAE:  6.9206 , Val Loss:  93.4082
```



☐ Show data download links
☑ Ignore outliers in chart scaling

Tooltip sorting method: default

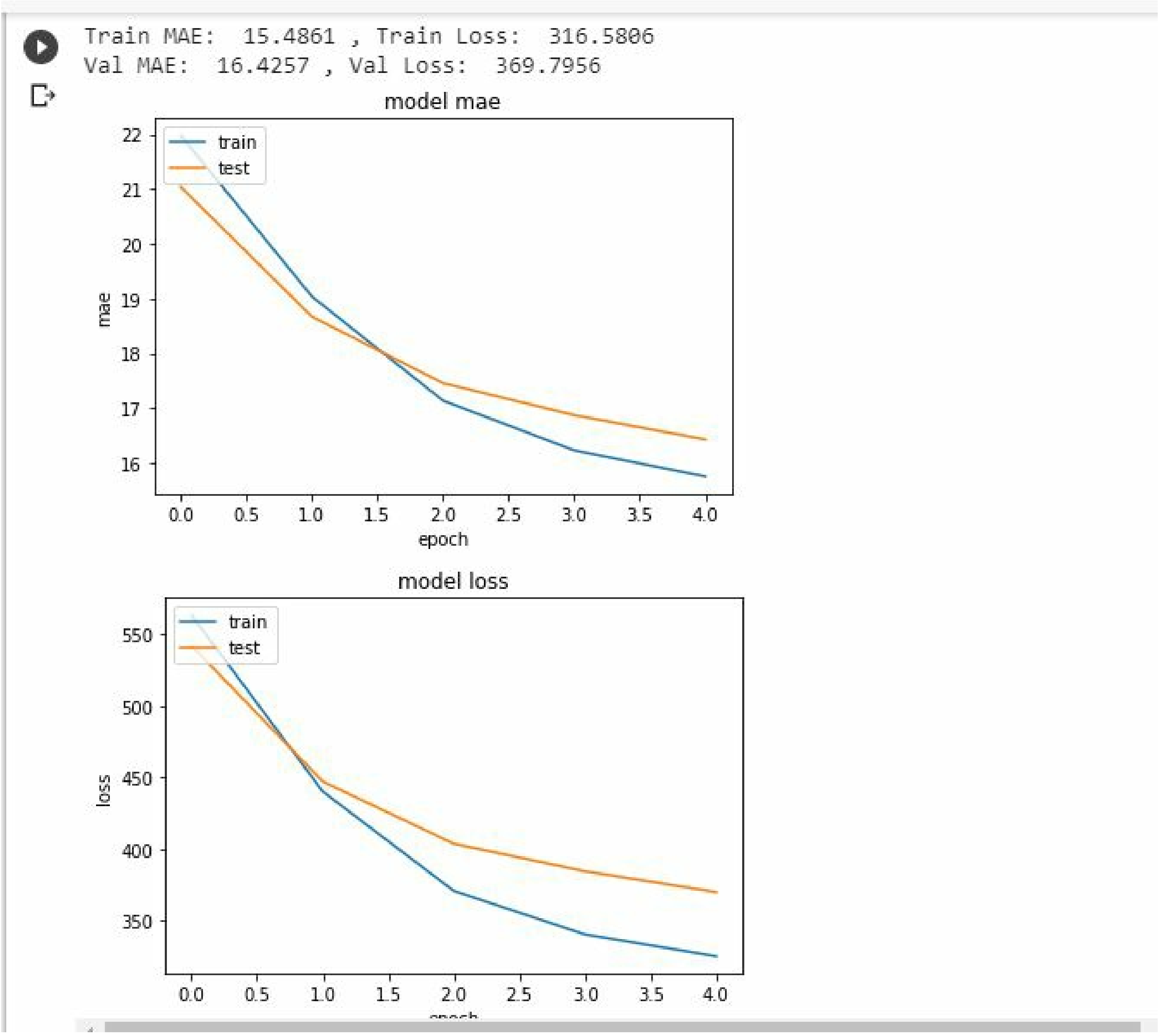
Smoothing: 0.6

Horizontal Axis: STEP RELATIVE WALL

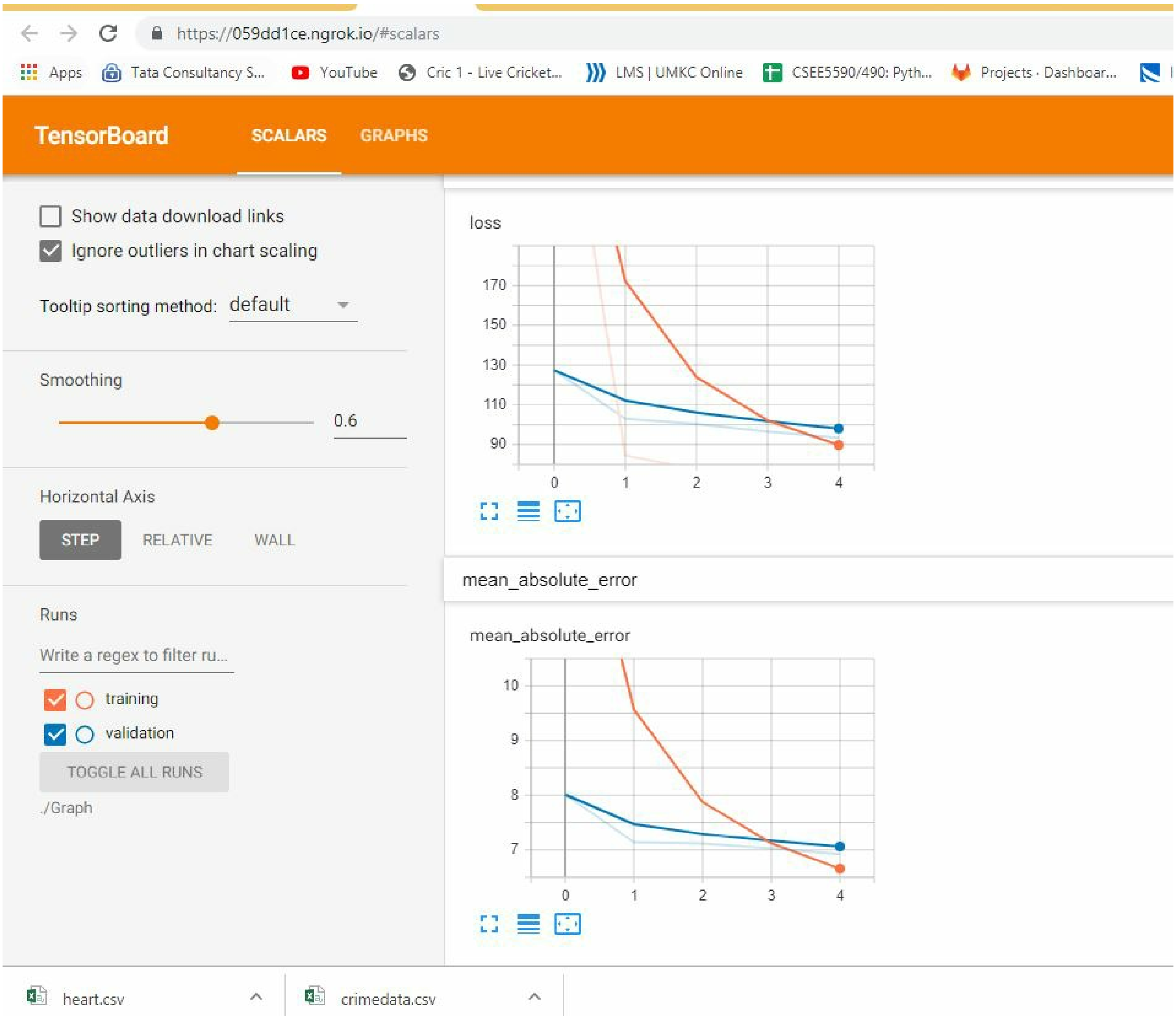
Runs

☑ training
☑ validation

after changing parameters



```
Train MAE:  15.4861 , Train Loss:  316.5806
Val MAE:  16.4257 , Val Loss:  369.7956
```



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

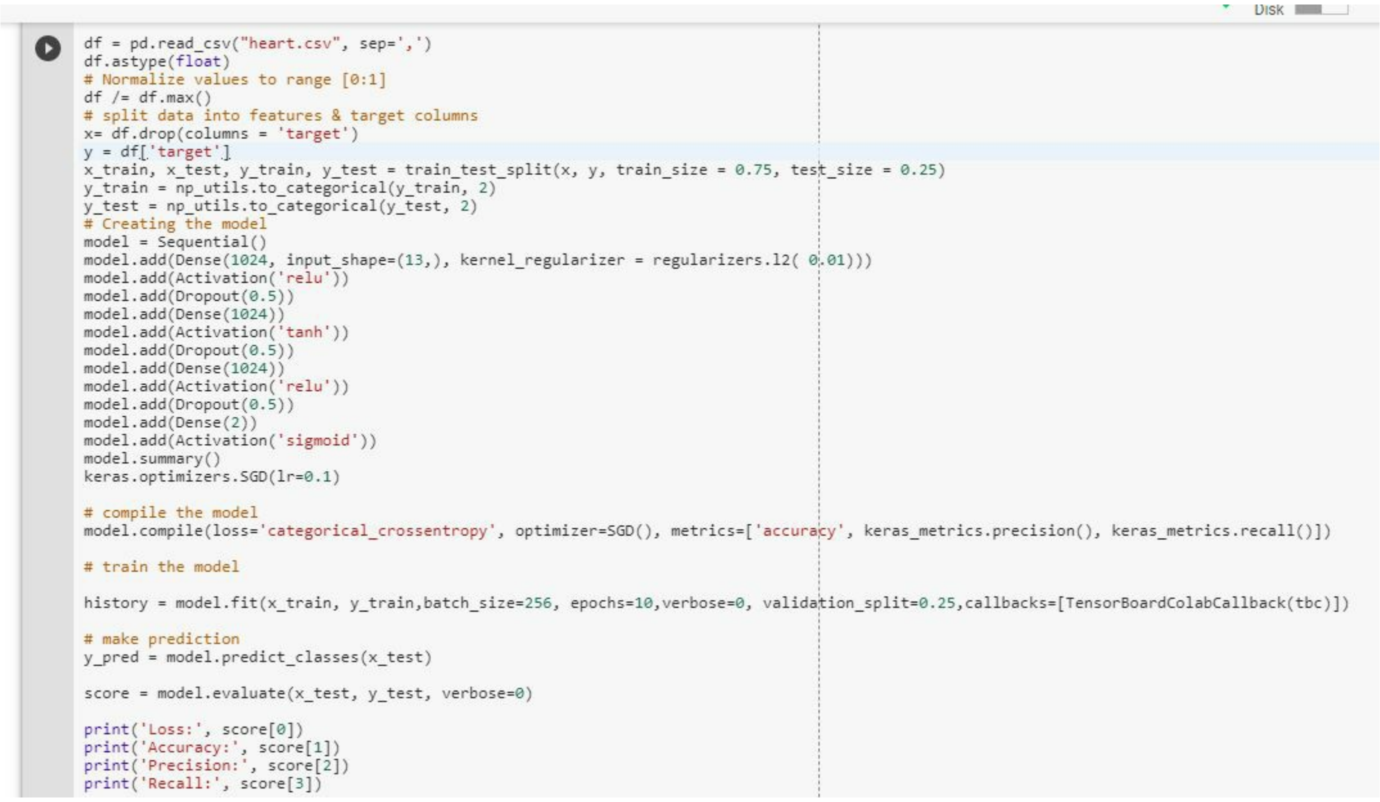
Write a regex to filter ru...

☑ ○ training

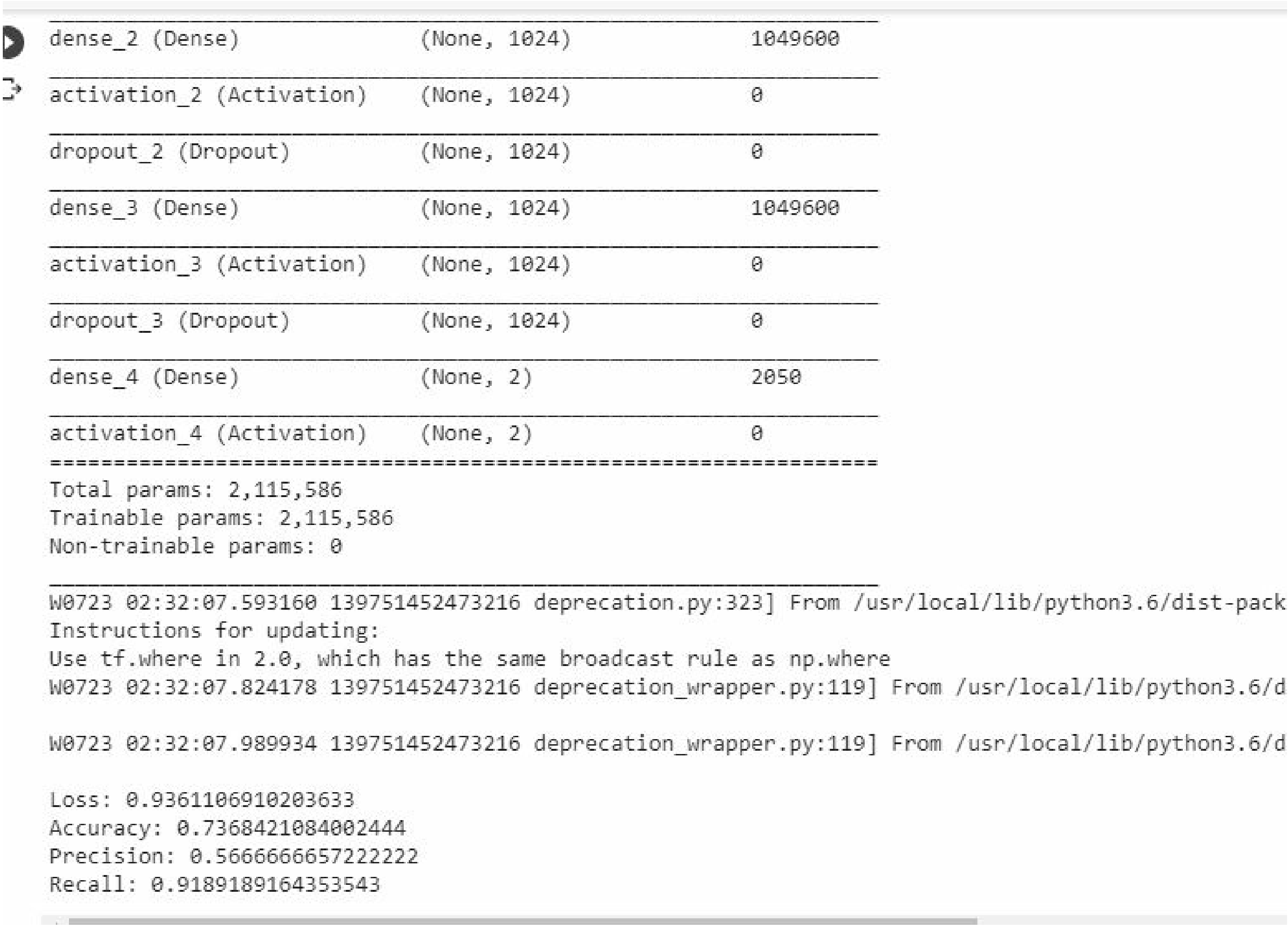
☑ ○ validation

2. We have taken Heart where ‘target’ column is the target column depends on other features in the data. In the first instant We have used SGD optimizer,learning rate = 0.1,batch size = 256 and sigmoid activation function. In the second instant We have used Adam optimizer,learning rate = 0.2,batch size = 128 and tanh activation function. The accuracy has decreased. It may be one of the factor that learning rate has increased so that model tries to train fast and accuracy decreases. The other factors are change in activation function, batch size and optimizer.

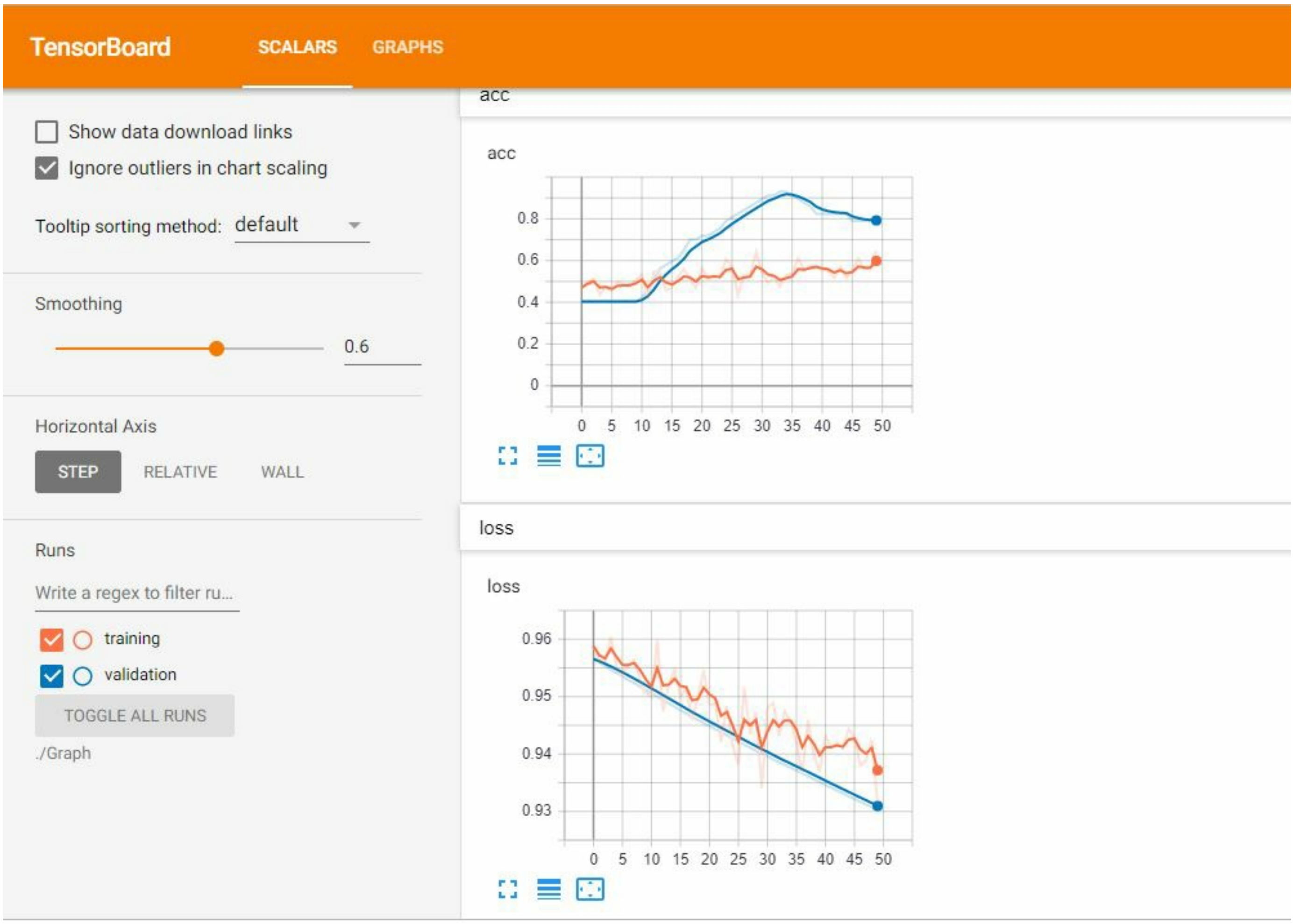
Please find the screenshots below



```
df = pd.read_csv("heart.csv", sep=',')
df.astype(float)
# Normalize values to range [0:1]
df /= df.max()
# split data into features & target columns
x= df.drop(columns = 'target')
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.75, test_size = 0.25)
y_train = np_utils.to_categorical(y_train, 2)
y_test = np_utils.to_categorical(y_test, 2)
# Creating the model
model = Sequential()
model.add(Dense(1024, input_shape=(13,), kernel_regularizer = regularizers.l2( 0.01)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024))
model.add(Activation('tanh'))
model.add(Dropout(0.5))
model.add(Dense(1024))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(2))
model.add(Activation('sigmoid'))
model.summary()
keras.optimizers.SGD(lr=0.1)

# compile the model
model.compile(loss='categorical_crossentropy', optimizer=SGD(), metrics=['accuracy', keras_metrics.precision(), keras_metrics.recall()])

# train the model

history = model.fit(x_train, y_train,batch_size=256, epochs=10,verbose=0, validation_split=0.25,callbacks=[TensorBoardColabCallback(tbc)])

# make prediction
y_pred = model.predict_classes(x_test)

score = model.evaluate(x_test, y_test, verbose=0)

print('Loss:', score[0])
print('Accuracy:', score[1])
print('Precision:', score[2])
print('Recall:', score[3])
```



```
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
activation_2 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              1049600
_________________________________________________________________
activation_3 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 2050
_________________________________________________________________
activation_4 (Activation)    (None, 2)                 0
=================================================================
Total params: 2,115,586
Trainable params: 2,115,586
Non-trainable params: 0
_________________________________________________________________
W0723 02:32:07.593160 139751452473216 deprecation.py:323] From /usr/local/lib/python3.6/dist-pack
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W0723 02:32:07.824178 139751452473216 deprecation_wrapper.py:119] From /usr/local/lib/python3.6/d

W0723 02:32:07.989934 139751452473216 deprecation_wrapper.py:119] From /usr/local/lib/python3.6/d

Loss: 0.9361106910203633
Accuracy: 0.7368421084002444
Precision: 0.5666666657222222
Recall: 0.9189189164353543
```



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter ru...

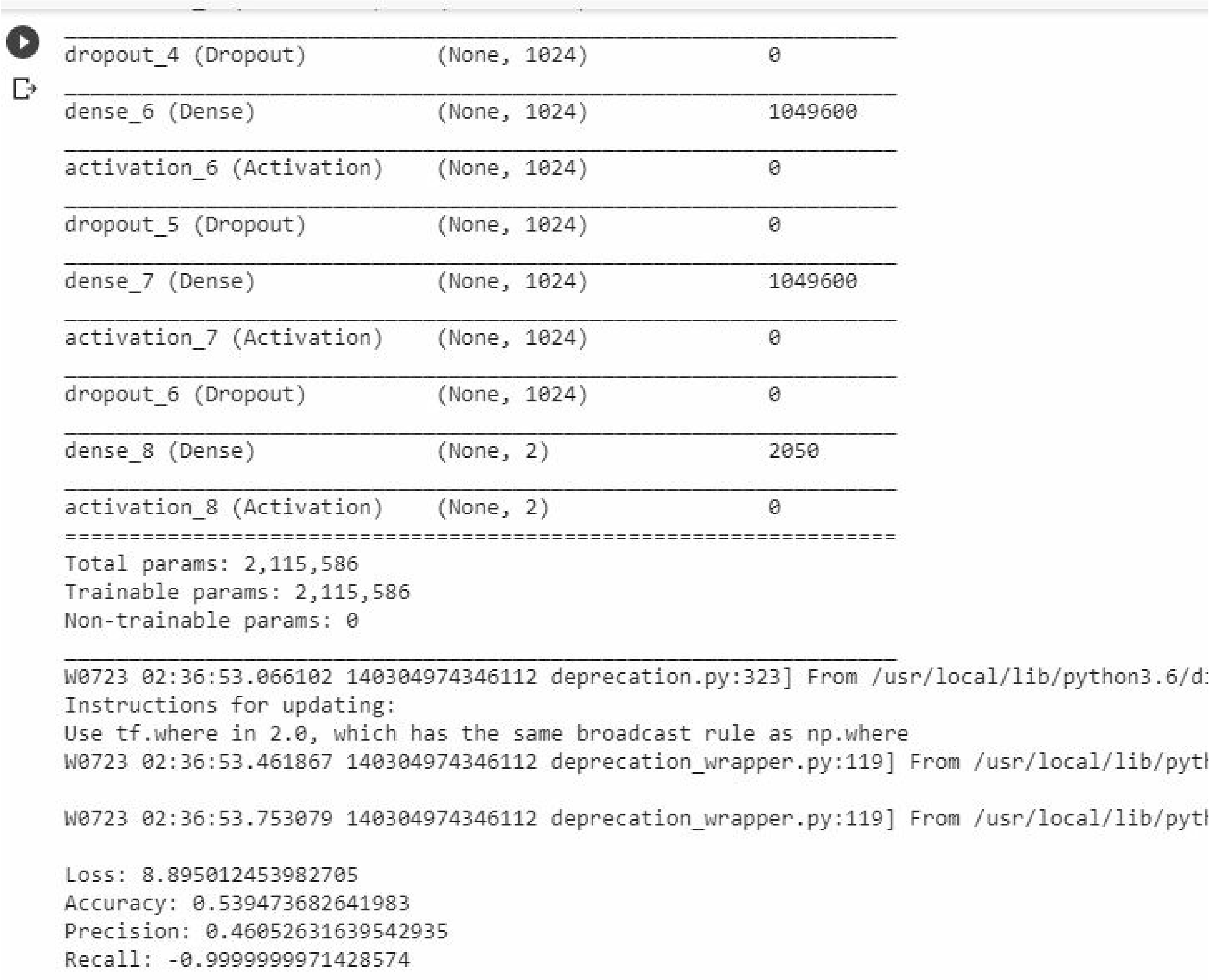
☑ ○ training

☑ ○ validation

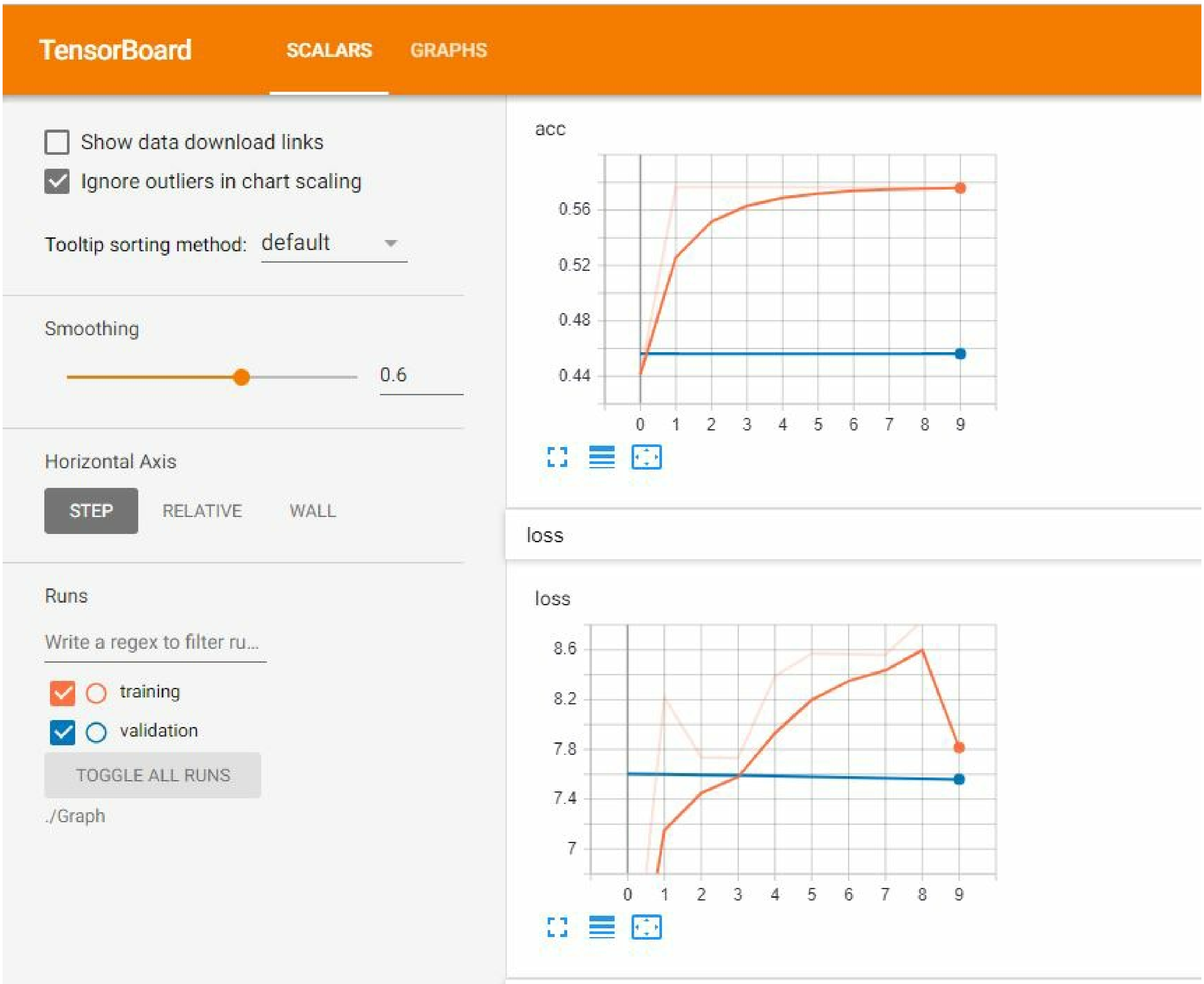
TOGGLE ALL RUNS

./Graph

after changing parameters



```
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_6 (Dense)              (None, 1024)              1049600
_________________________________________________________________
activation_6 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_5 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              1049600
_________________________________________________________________
activation_7 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 2050
_________________________________________________________________
activation_8 (Activation)    (None, 2)                 0
=================================================================
Total params: 2,115,586
Trainable params: 2,115,586
Non-trainable params: 0
_________________________________________________________________
W0723 02:36:53.066102 140304974346112 deprecation.py:323] From /usr/local/lib/python3.6/d
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W0723 02:36:53.461867 140304974346112 deprecation_wrapper.py:119] From /usr/local/lib/pyt

W0723 02:36:53.753079 140304974346112 deprecation_wrapper.py:119] From /usr/local/lib/pyt

Loss: 8.895012453982705
Accuracy: 0.539473682641983
Precision: 0.46052631639542935
Recall: -0.9999999971428574
```



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter ru...

☑ ○ training

☑ ○ validation

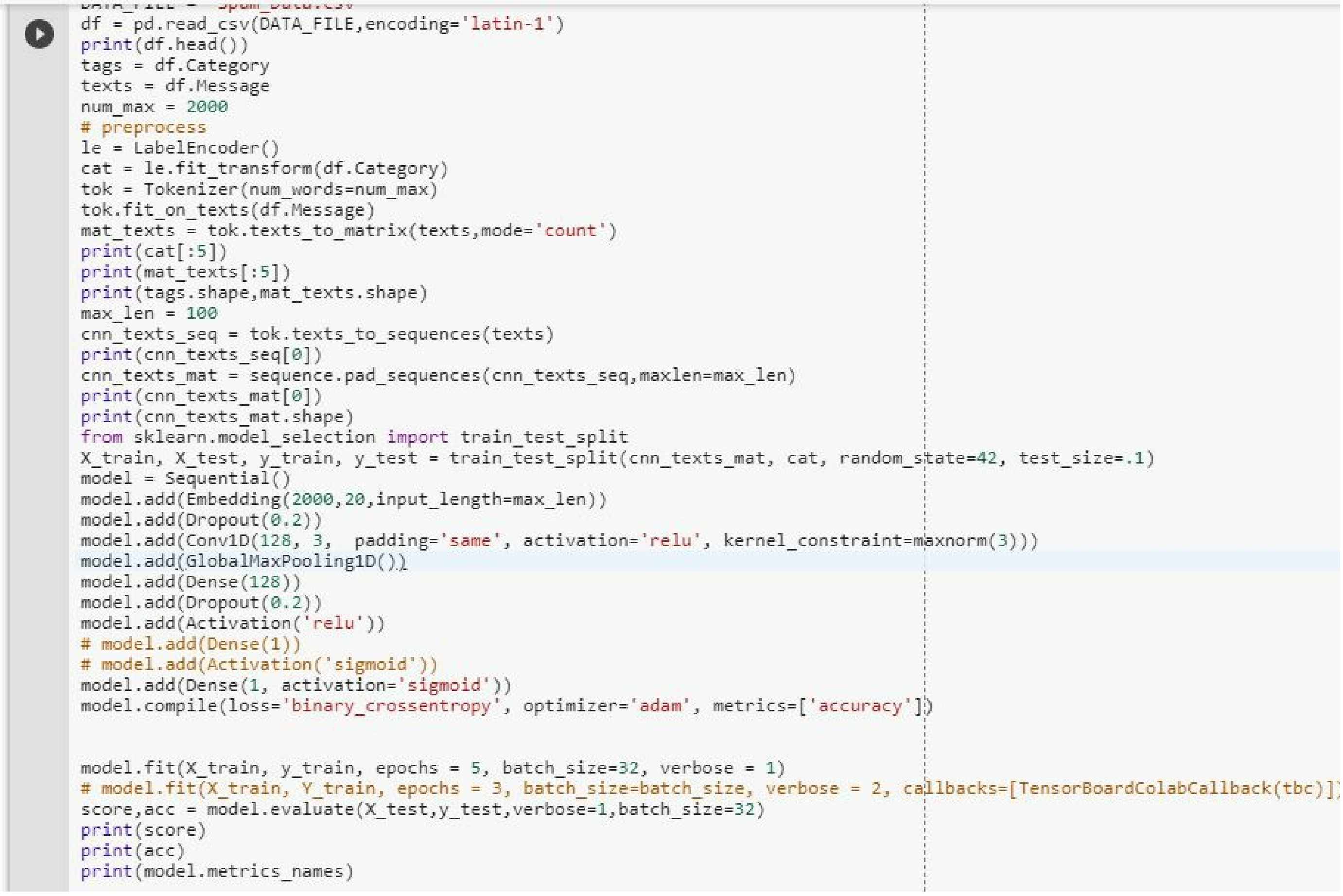
./Graph

3. We have taken Spam where ‘category’ column is the target column depends on text feature in the data using CNN Creating Embedding layer as the input layer with 2000 feature set, input length as the max length of a sentence and 20% of dropouts. Using relu activation function in the Convolution layer of single dimension with 128 neurons with 3 one dimension filter. Adding MaxPooling layer with 2 pool size. Added one more relu activation function Convolution layer with 128 neurons with a feature detector of 2. Adding MaxPooling layer with 2 pool size. Flattening to the single vector. Adding the output layer with sigmoid activation function as there are only 2 categories

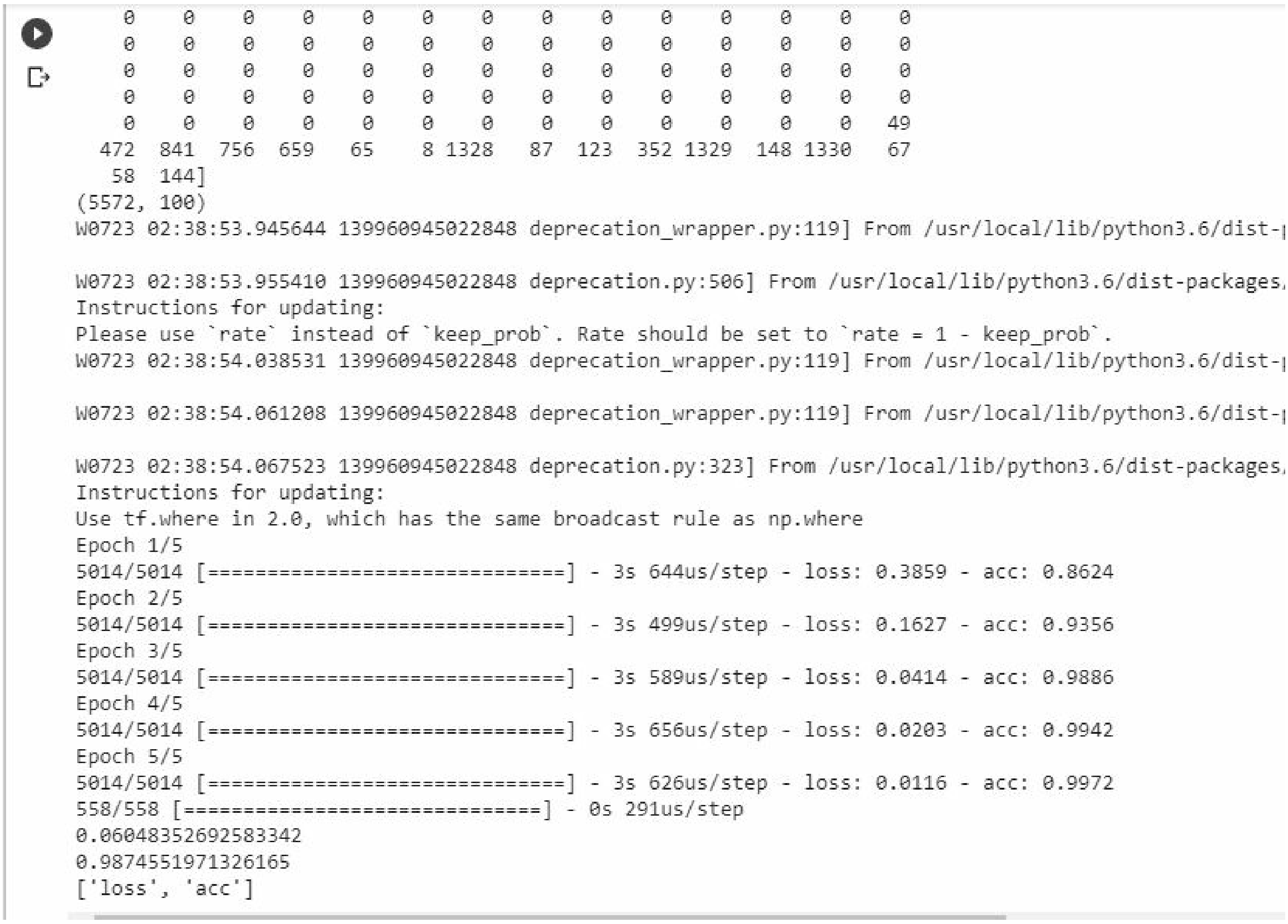
(binary Classification).

After building the model - running the model with 5 epochs, 32 batch size.

Checking the train data accuracy, score and the validation data accuracy, score



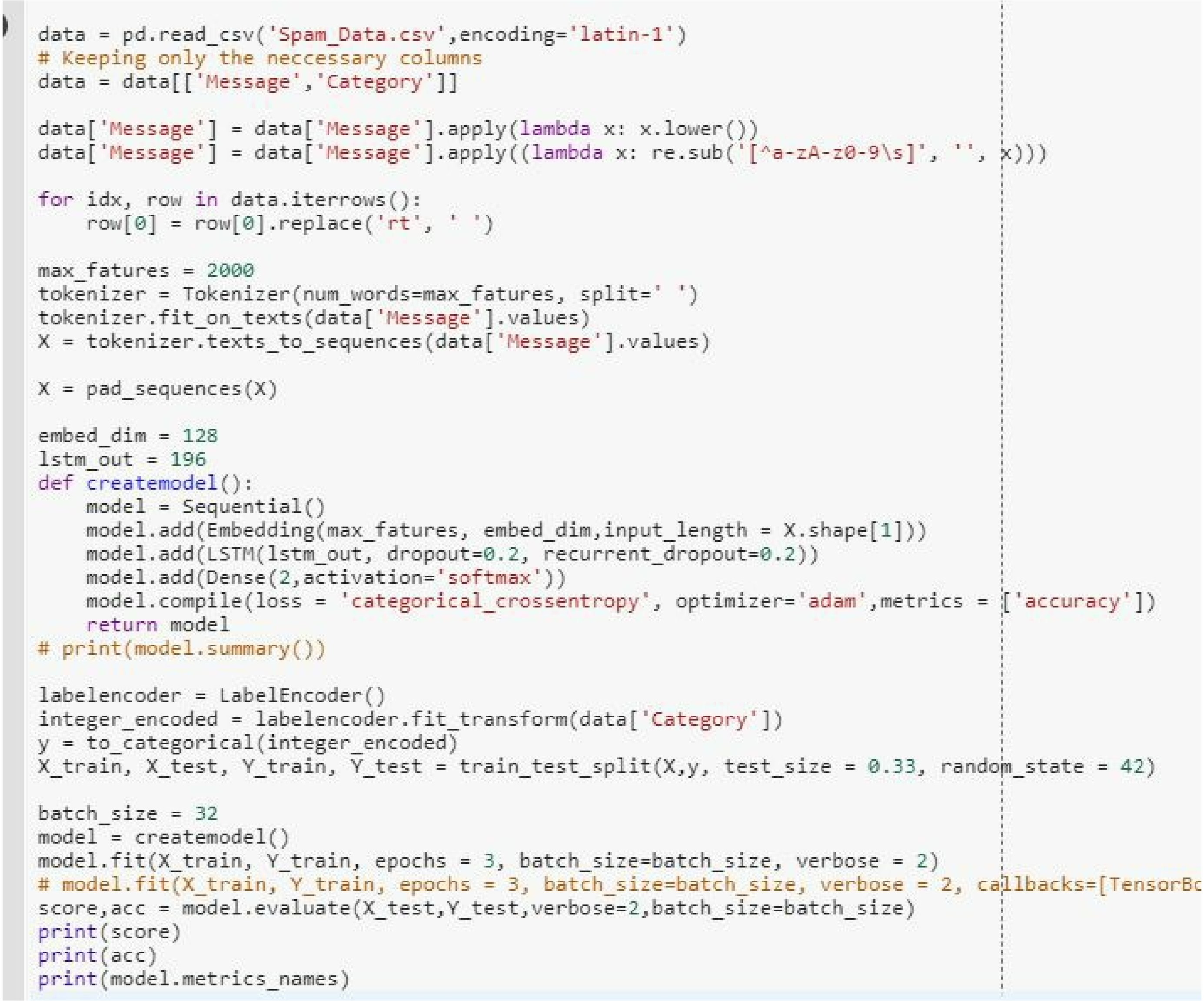
```
DATA_FILE = 'Spam_Data.csv'
df = pd.read_csv(DATA_FILE,encoding='latin-1')
print(df.head())
tags = df.Category
texts = df.Message
num_max = 2000
# preprocess
le = LabelEncoder()
cat = le.fit_transform(df.Category)
tok = Tokenizer(num_words=num_max)
tok.fit_on_texts(df.Message)
mat_texts = tok.texts_to_matrix(texts,mode='count')
print(cat[:5])
print(mat_texts[:5])
print(tags.shape,mat_texts.shape)
max_len = 100
cnn_texts_seq = tok.texts_to_sequences(texts)
print(cnn_texts_seq[0])
cnn_texts_mat = sequence.pad_sequences(cnn_texts_seq,maxlen=max_len)
print(cnn_texts_mat[0])
print(cnn_texts_mat.shape)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(cnn_texts_mat, cat, random_state=42, test_size=.1)
model = Sequential()
model.add(Embedding(2000,20,input_length=max_len))
model.add(Dropout(0.2))
model.add(Conv1D(128, 3,  padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(GlobalMaxPooling1D())
model.add(Dense(128))
model.add(Dropout(0.2))
model.add(Activation('relu'))
# model.add(Dense(1))
# model.add(Activation('sigmoid'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


model.fit(X_train, y_train, epochs = 5, batch_size=32, verbose = 1)
# model.fit(X_train, Y_train, epochs = 3, batch_size=batch_size, verbose = 2, callbacks=[TensorBoardColabCallback(tbc)]
score,acc = model.evaluate(X_test,y_test,verbose=1,batch_size=32)
print(score)
print(acc)
print(model.metrics_names)
```



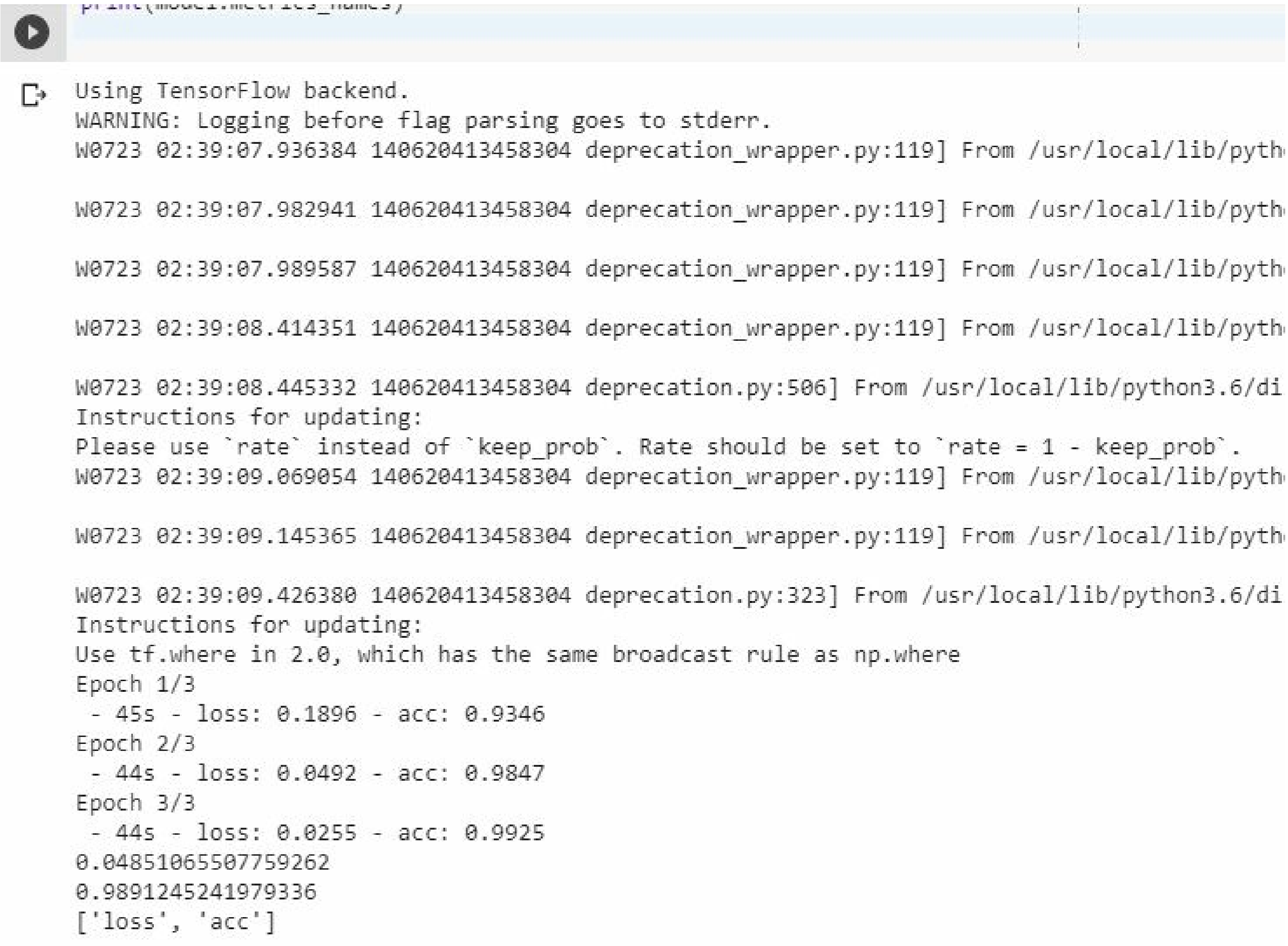
```
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0   49
   472  841  756  659   65    8 1328   87  123  352 1329  148 1330   67
    58  144]
(5572, 100)
W0723 02:38:53.945644 139960945022848 deprecation_wrapper.py:119] From /usr/local/lib/python3.6/dist-

W0723 02:38:53.955410 139960945022848 deprecation.py:506] From /usr/local/lib/python3.6/dist-packages,
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
W0723 02:38:54.038531 139960945022848 deprecation_wrapper.py:119] From /usr/local/lib/python3.6/dist-

W0723 02:38:54.061208 139960945022848 deprecation_wrapper.py:119] From /usr/local/lib/python3.6/dist-

W0723 02:38:54.067523 139960945022848 deprecation.py:323] From /usr/local/lib/python3.6/dist-packages,
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/5
5014/5014 [==============================] - 3s 644us/step - loss: 0.3859 - acc: 0.8624
Epoch 2/5
5014/5014 [==============================] - 3s 499us/step - loss: 0.1627 - acc: 0.9356
Epoch 3/5
5014/5014 [==============================] - 3s 589us/step - loss: 0.0414 - acc: 0.9886
Epoch 4/5
5014/5014 [==============================] - 3s 656us/step - loss: 0.0203 - acc: 0.9942
Epoch 5/5
5014/5014 [==============================] - 3s 626us/step - loss: 0.0116 - acc: 0.9972
558/558 [==============================] - 0s 291us/step
0.06048352692583342
0.9874551971326165
['loss', 'acc']
```

4. We have done analysis on the same dataset using LSTM model. Created a model and built with LSTM layer. Checking the train data accuracy, score and the validation data accuracy, score

Please find the screenshots below



```
data = pd.read_csv('Spam_Data.csv',encoding='latin-1')
# Keeping only the neccessary columns
data = data[['Message','Category']]

data['Message'] = data['Message'].apply(lambda x: x.lower())
data['Message'] = data['Message'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

for idx, row in data.iterrows():
    row[0] = row[0].replace('rt', ' ')

max_fatures = 2000
tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(data['Message'].values)
X = tokenizer.texts_to_sequences(data['Message'].values)

X = pad_sequences(X)

embed_dim = 128
lstm_out = 196
def createmodel():
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(2,activation='softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
    return model
# print(model.summary())

labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['Category'])
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)

batch_size = 32
model = createmodel()
model.fit(X_train, Y_train, epochs = 3, batch_size=batch_size, verbose = 2)
# model.fit(X_train, Y_train, epochs = 3, batch_size=batch_size, verbose = 2, callbacks=[TensorBo
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)
print(score)
print(acc)
print(model.metrics_names)
```



```
print(model.metrics_names)
```

```
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0723 02:39:07.936384 140620413458304 deprecation_wrapper.py:119] From /usr/local/lib/pyth

W0723 02:39:07.982941 140620413458304 deprecation_wrapper.py:119] From /usr/local/lib/pyth

W0723 02:39:07.989587 140620413458304 deprecation_wrapper.py:119] From /usr/local/lib/pyth

W0723 02:39:08.414351 140620413458304 deprecation_wrapper.py:119] From /usr/local/lib/pyth

W0723 02:39:08.445332 140620413458304 deprecation.py:506] From /usr/local/lib/python3.6/di
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
W0723 02:39:09.069054 140620413458304 deprecation_wrapper.py:119] From /usr/local/lib/pyth

W0723 02:39:09.145365 140620413458304 deprecation_wrapper.py:119] From /usr/local/lib/pyth

W0723 02:39:09.426380 140620413458304 deprecation.py:323] From /usr/local/lib/python3.6/di
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/3
 - 45s - loss: 0.1896 - acc: 0.9346
Epoch 2/3
 - 44s - loss: 0.0492 - acc: 0.9847
Epoch 3/3
 - 44s - loss: 0.0255 - acc: 0.9925
0.04851065507759262
0.9891245241979336
['loss', 'acc']
```

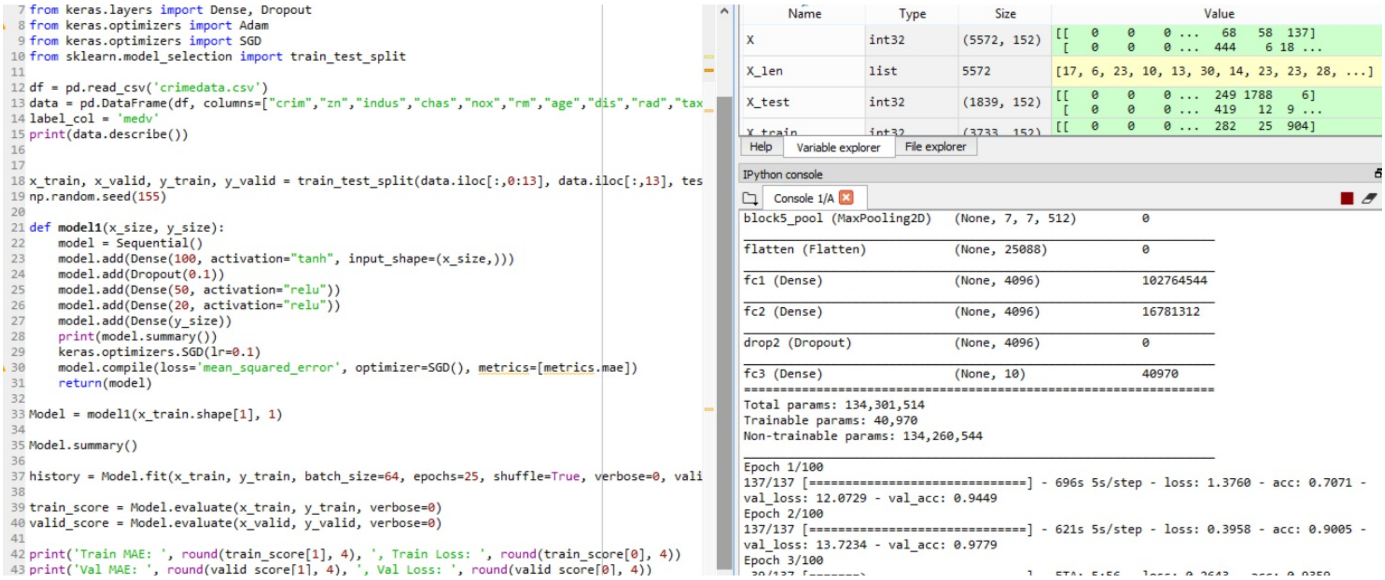
5. CNN shows more better results compared to LSTM model in case of text classification with large set of data where as in case of small data set, the accuracies and the loss functions are almost same.

   The accuracies are more than 98% in case of spam dataset.

6. We have taken monkey species dataset. I am unable to run it in Google collab as dataset is huge to upload to drive. Dataset contains images in the training folder and validation folder. After removing the stopwords tokenized and stemmed with Snowball-Stemmer. Both the datasets after preprocessing are stored in the dictionary. words are converted into the vectors after tokenizing.

CNN model is created and used softmax as the activation function for the last layer. optimizer: Adam() batch size: 256 epochs: 5 Model is evaluated for accuracy.

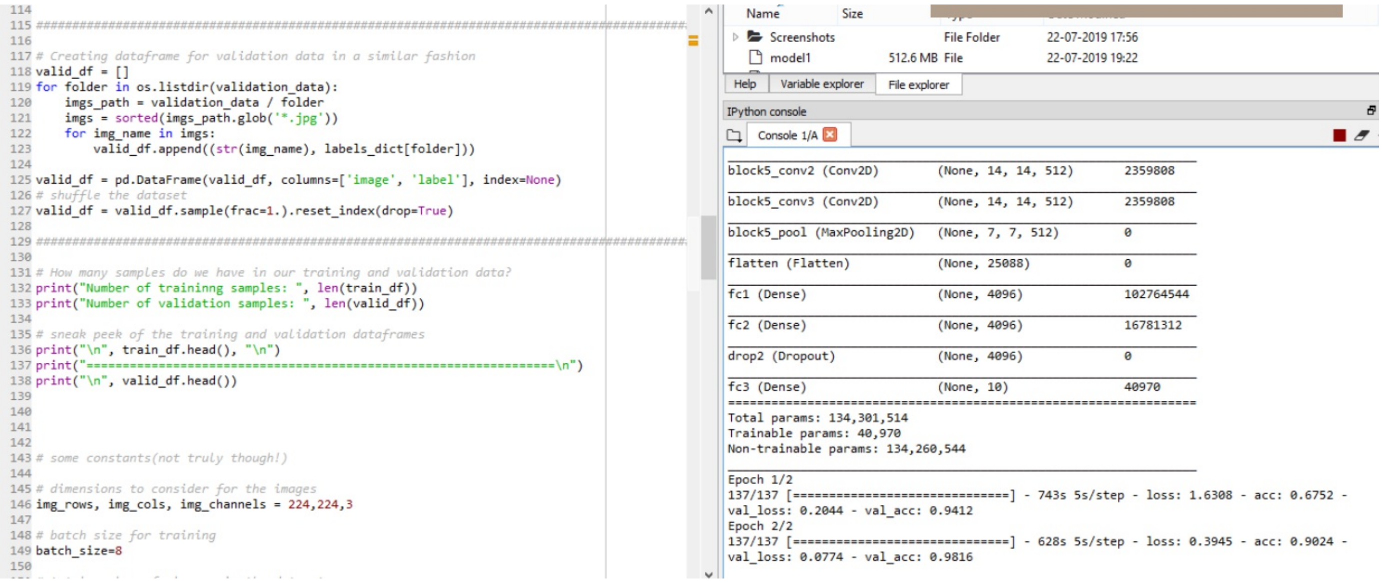
Please find the screenshots below



```python
from keras.layers import Dense, Dropout
from keras.optimizers import Adam
from keras.optimizers import SGD
from sklearn.model_selection import train_test_split

df = pd.read_csv('crimedata.csv')
data = pd.DataFrame(df, columns=["crim","zn","indus","chas","nox","rm","age","dis","rad","tax
label_col = 'medv'
print(data.describe())


x_train, x_valid, y_train, y_valid = train_test_split(data.iloc[:,0:13], data.iloc[:,13], tes
np.random.seed(155)

def model1(x_size, y_size):
    model = Sequential()
    model.add(Dense(100, activation="tanh", input_shape=(x_size,)))
    model.add(Dropout(0.1))
    model.add(Dense(50, activation="relu"))
    model.add(Dense(20, activation="relu"))
    model.add(Dense(y_size))
    print(model.summary())
    keras.optimizers.SGD(lr=0.1)
    model.compile(loss='mean_squared_error', optimizer=SGD(), metrics=[metrics.mae])
    return(model)

Model = model1(x_train.shape[1], 1)

Model.summary()

history = Model.fit(x_train, y_train, batch_size=64, epochs=25, shuffle=True, verbose=0, vali

train_score = Model.evaluate(x_train, y_train, verbose=0)
valid_score = Model.evaluate(x_valid, y_valid, verbose=0)

print('Train MAE: ', round(train_score[1], 4), ', Train Loss: ', round(train_score[0], 4))
print('Val MAE: ', round(valid score[1], 4), ', Val Loss: ', round(valid score[0], 4))
```

Name	Type	Size	Value
X	int32	(5572, 152)	[[0 0 0 ... 68 58 137] [0 0 0 ... 444 6 18 ...
X_len	list	5572	[17, 6, 23, 10, 13, 30, 14, 23, 23, 28, ...]
X_test	int32	(1839, 152)	[[0 0 0 ... 249 1788 6] [0 0 0 ... 419 12 9 ...
X_train	int32	(3733, 152)	[[0 0 0 ... 282 25 904]

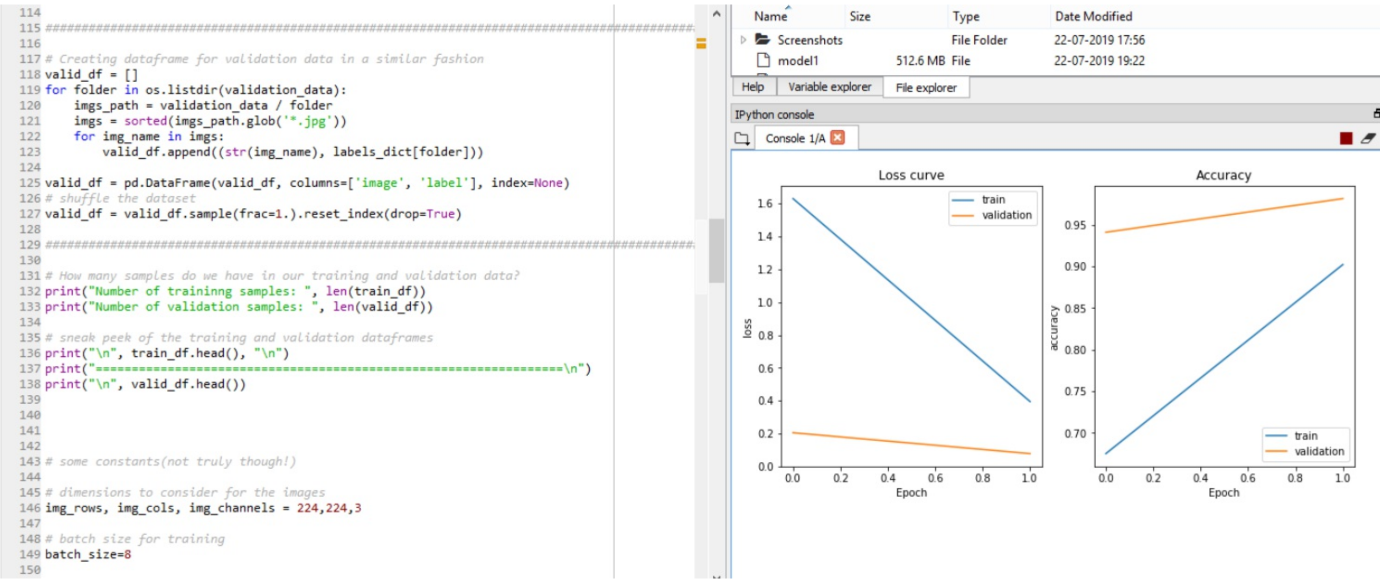
```
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
drop2 (Dropout)              (None, 4096)              0
_________________________________________________________________
fc3 (Dense)                  (None, 10)                40970
=================================================================
Total params: 134,301,514
Trainable params: 40,970
Non-trainable params: 134,260,544
_________________________________________________________________
Epoch 1/100
137/137 [==============================] - 696s 5s/step - loss: 1.3760 - acc: 0.7071 -
val_loss: 12.0729 - val_acc: 0.9449
Epoch 2/100
137/137 [==============================] - 621s 5s/step - loss: 0.3958 - acc: 0.9005 -
val_loss: 13.7234 - val_acc: 0.9779
Epoch 3/100
 30/137 [=======>                      ] - ETA: 5:56 - loss: 0.2643 - acc: 0.9359
```



```python

################################################################################

# Creating dataframe for validation data in a similar fashion
valid_df = []
for folder in os.listdir(validation_data):
    imgs_path = validation_data / folder
    imgs = sorted(imgs_path.glob('*.jpg'))
    for img_name in imgs:
        valid_df.append((str(img_name), labels_dict[folder]))

valid_df = pd.DataFrame(valid_df, columns=['image', 'label'], index=None)
# shuffle the dataset
valid_df = valid_df.sample(frac=1.).reset_index(drop=True)

################################################################################

# How many samples do we have in our training and validation data?
print("Number of traininng samples: ", len(train_df))
print("Number of validation samples: ", len(valid_df))

# sneak peek of the training and validation dataframes
print("\n", train_df.head(), "\n")
print("=================================================================\n")
print("\n", valid_df.head())




# some constants(not truly though!)

# dimensions to consider for the images
img_rows, img_cols, img_channels = 224,224,3

# batch size for training
batch_size=8

```

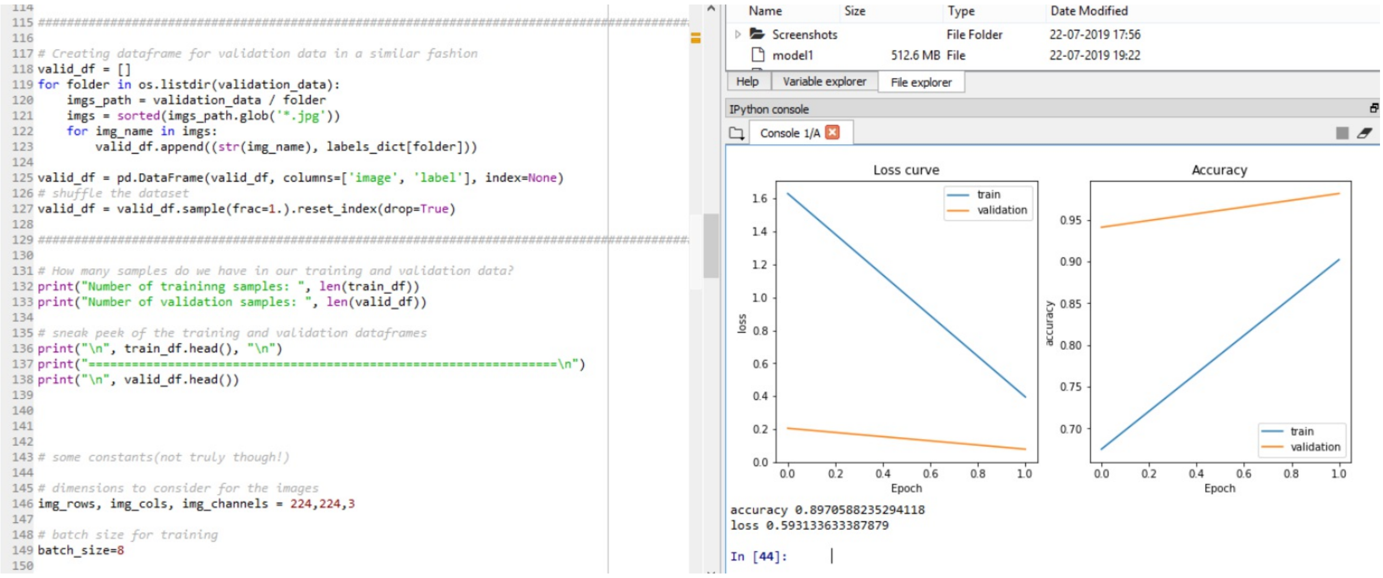
Name	Size	[redacted]	
Screenshots		File Folder	22-07-2019 17:56
model1	512.6 MB	File	22-07-2019 19:22

```
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
drop2 (Dropout)              (None, 4096)              0
_________________________________________________________________
fc3 (Dense)                  (None, 10)                40970
=================================================================
Total params: 134,301,514
Trainable params: 40,970
Non-trainable params: 134,260,544
_________________________________________________________________
Epoch 1/2
137/137 [==============================] - 743s 5s/step - loss: 1.6308 - acc: 0.6752 -
val_loss: 0.2044 - val_acc: 0.9412
Epoch 2/2
137/137 [==============================] - 628s 5s/step - loss: 0.3945 - acc: 0.9024 -
val_loss: 0.0774 - val_acc: 0.9816
```



```
##########################################################################################

# Creating dataframe for validation data in a similar fashion
valid_df = []
for folder in os.listdir(validation_data):
    imgs_path = validation_data / folder
    imgs = sorted(imgs_path.glob('*.jpg'))
    for img_name in imgs:
        valid_df.append((str(img_name), labels_dict[folder]))

valid_df = pd.DataFrame(valid_df, columns=['image', 'label'], index=None)
# shuffle the dataset
valid_df = valid_df.sample(frac=1.).reset_index(drop=True)

##########################################################################################

# How many samples do we have in our training and validation data?
print("Number of traininng samples: ", len(train_df))
print("Number of validation samples: ", len(valid_df))

# sneak peek of the training and validation dataframes
print("\n", train_df.head(), "\n")
print("================================================================\n")
print("\n", valid_df.head())



# some constants(not truly though!)

# dimensions to consider for the images
img_rows, img_cols, img_channels = 224,224,3

# batch size for training
batch_size=8
```

Name	Size	Type	Date Modified
Screenshots		File Folder	22-07-2019 17:56
model1	512.6 MB	File	22-07-2019 19:22



```
##########################################################################################

# Creating dataframe for validation data in a similar fashion
valid_df = []
for folder in os.listdir(validation_data):
    imgs_path = validation_data / folder
    imgs = sorted(imgs_path.glob('*.jpg'))
    for img_name in imgs:
        valid_df.append((str(img_name), labels_dict[folder]))

valid_df = pd.DataFrame(valid_df, columns=['image', 'label'], index=None)
# shuffle the dataset
valid_df = valid_df.sample(frac=1.).reset_index(drop=True)

##########################################################################################

# How many samples do we have in our training and validation data?
print("Number of traininng samples: ", len(train_df))
print("Number of validation samples: ", len(valid_df))

# sneak peek of the training and validation dataframes
print("\n", train_df.head(), "\n")
print("================================================================\n")
print("\n", valid_df.head())



# some constants(not truly though!)

# dimensions to consider for the images
img_rows, img_cols, img_channels = 224,224,3

# batch size for training
batch_size=8
```

Name	Size	Type	Date Modified
Screenshots		File Folder	22-07-2019 17:56
model1	512.6 MB	File	22-07-2019 19:22

```
accuracy 0.8970588235294118
loss 0.593133633387879

In [44]:
```

Thank you