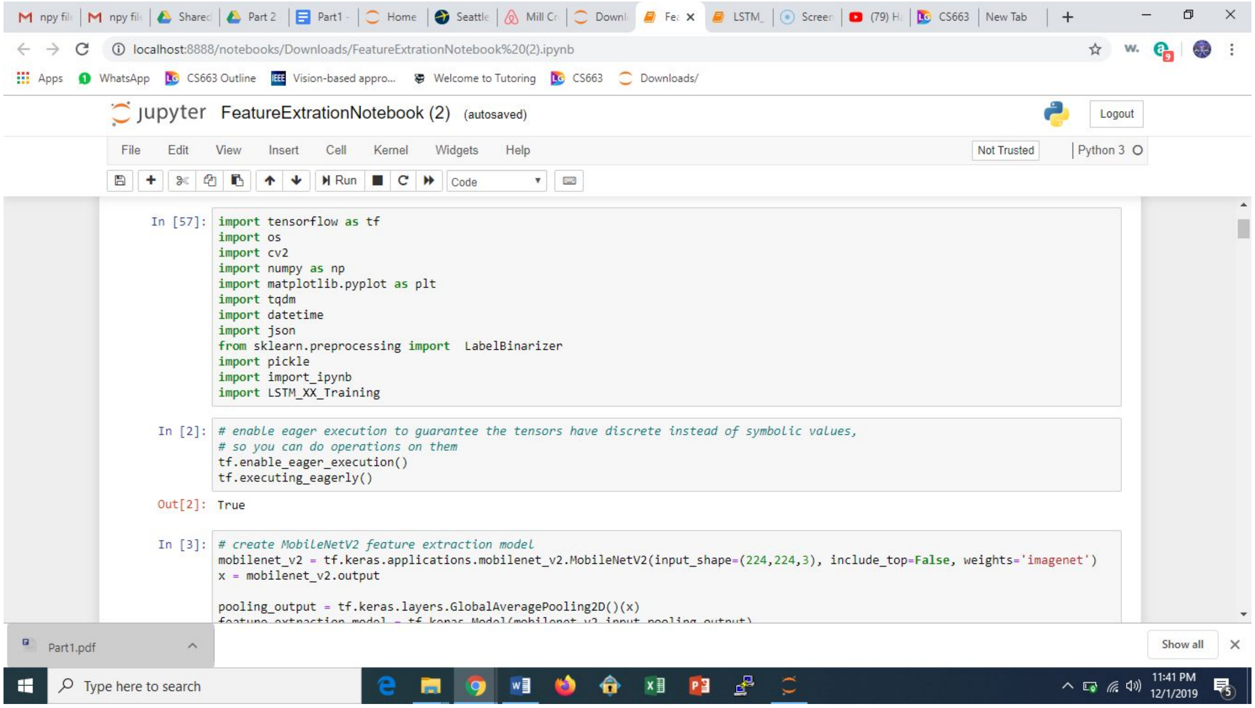
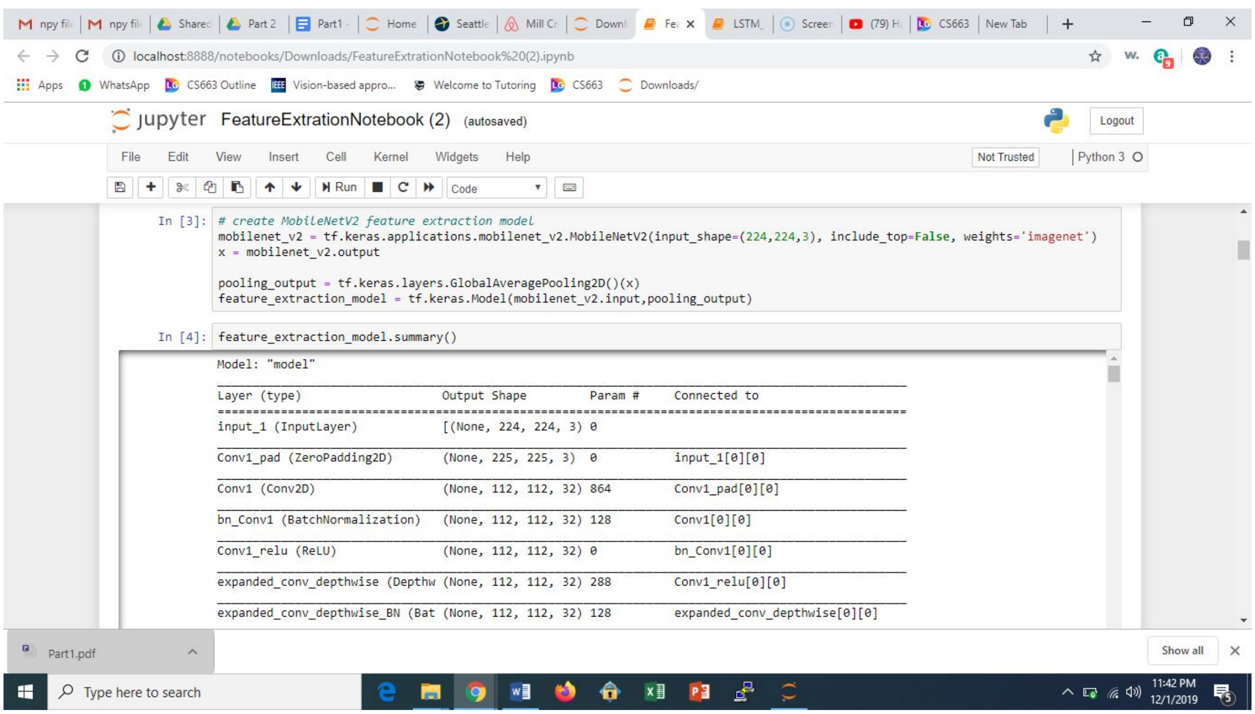
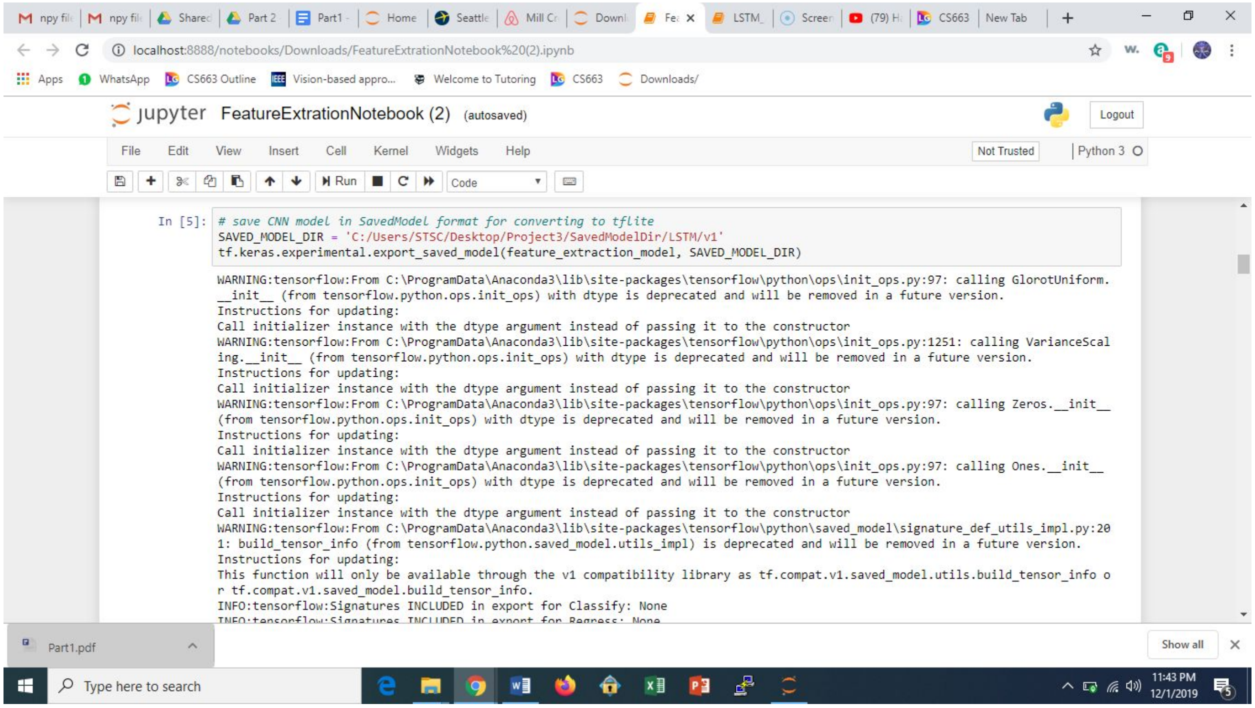


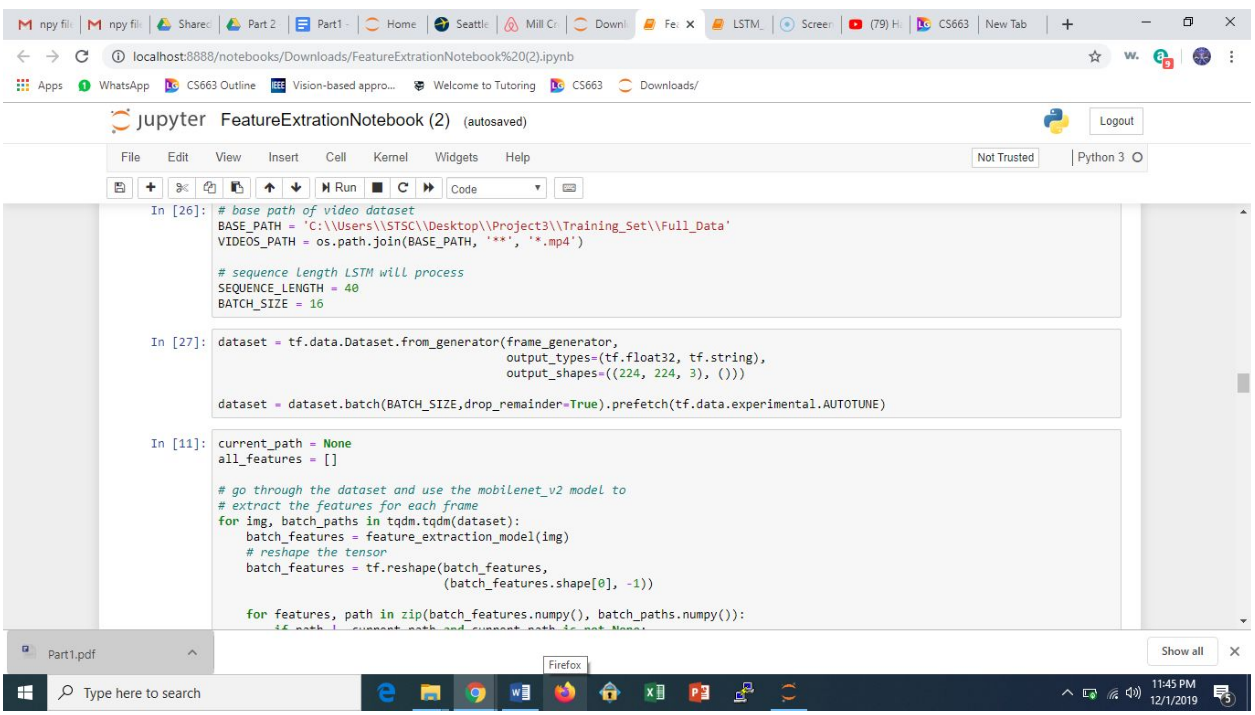
Import all dependencies



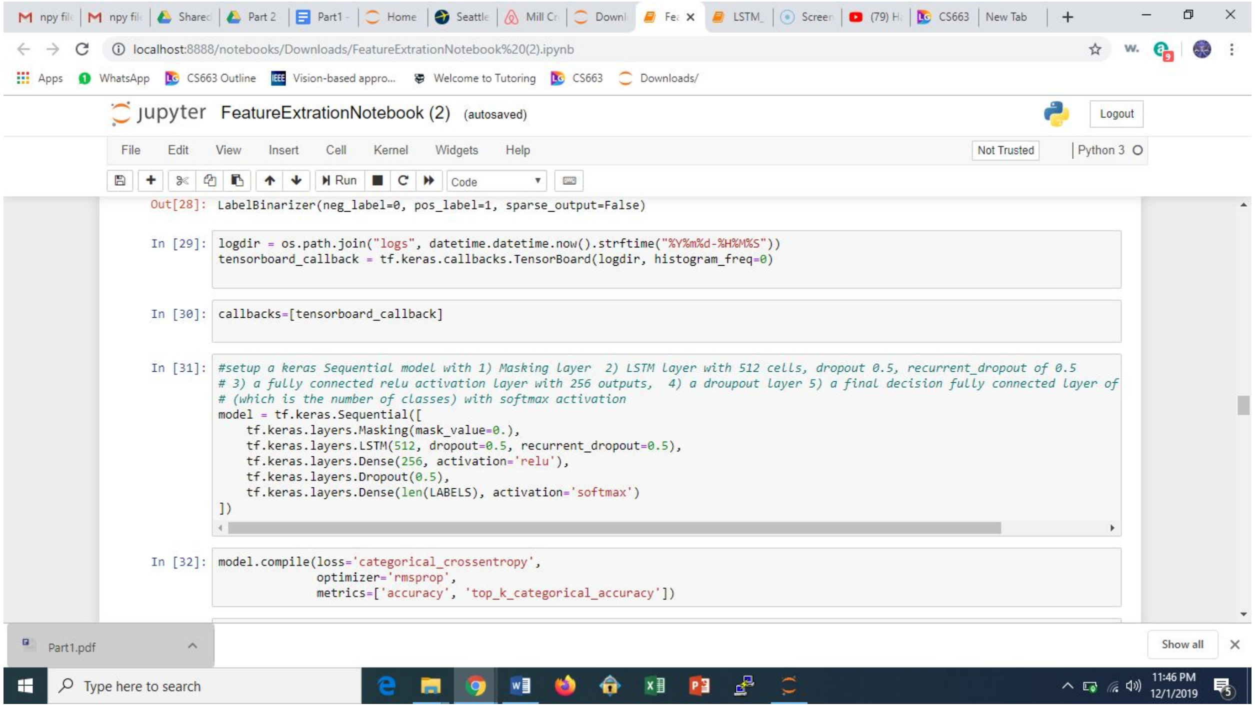
Create a feature extraction model



Save the feature extraction model

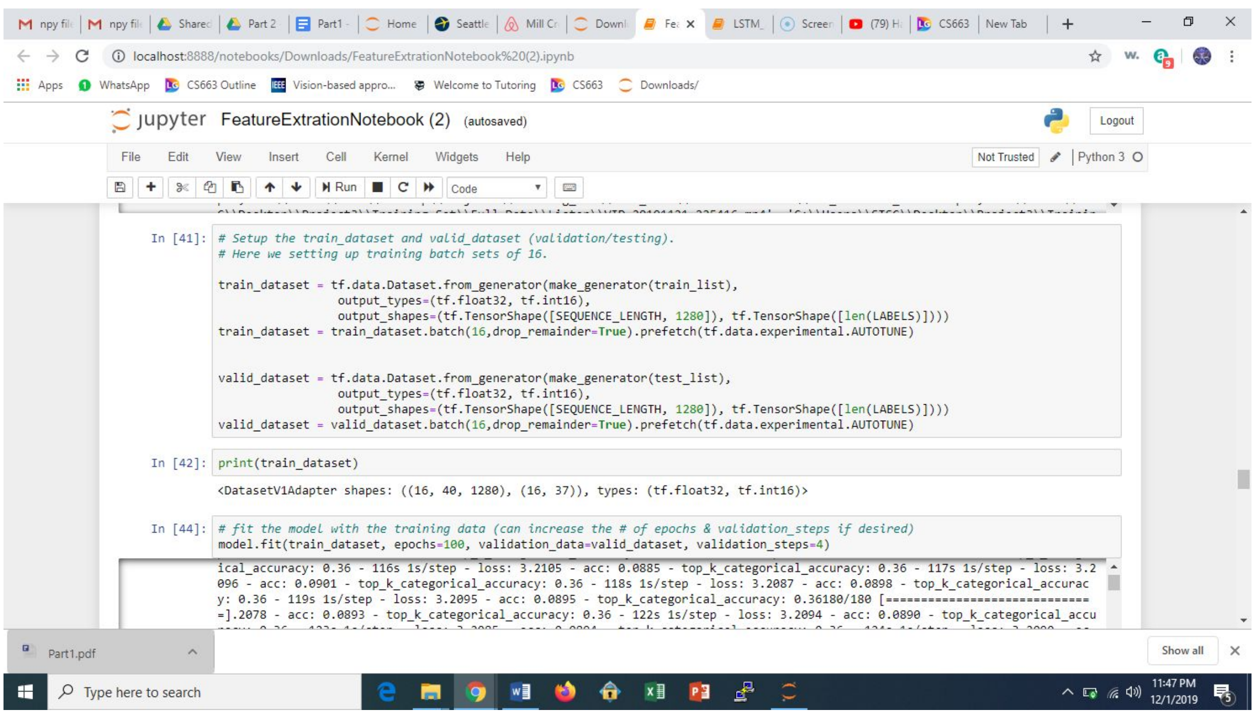


Set the path for video dataset and go through all videos and perform feature extraction



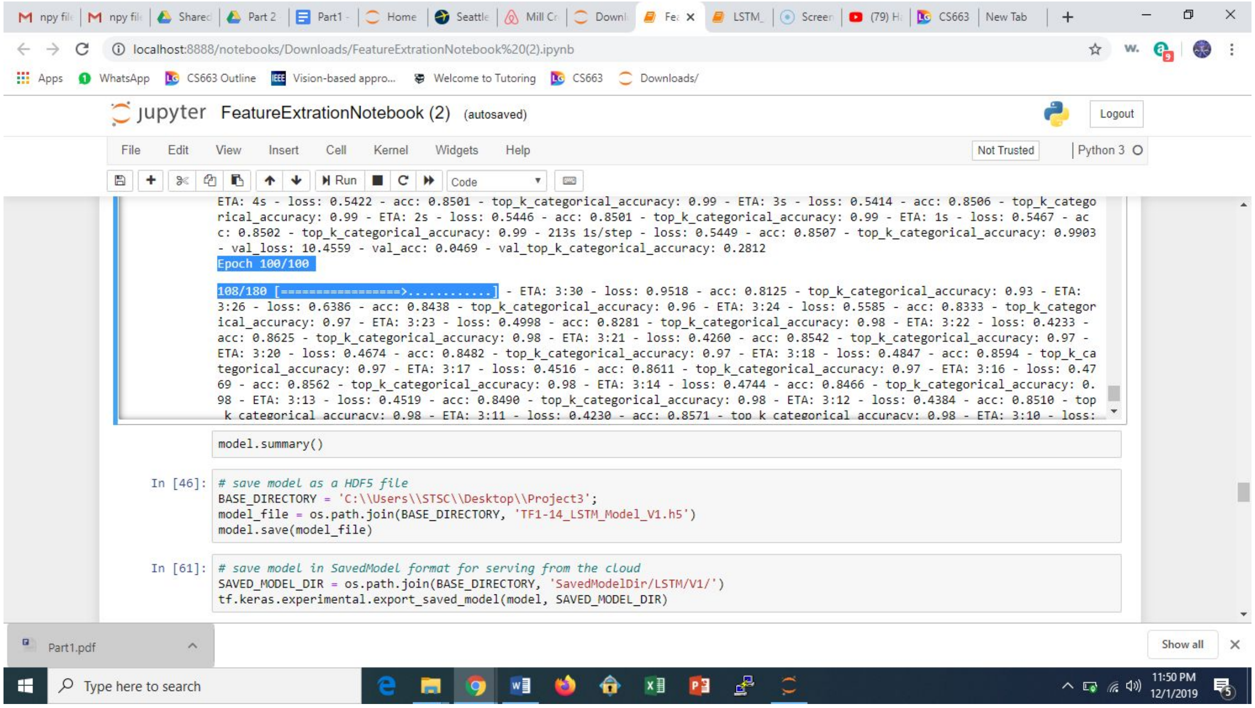

```
Out[28]: LabelBinarizer(neg_label=0, pos_label=1, sparse_output=False)

In [29]: logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
         tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=0)

In [30]: callbacks=[tensorboard_callback]

In [31]: #setup a keras Sequential model with 1) Masking layer  2) LSTM layer with 512 cells, dropout 0.5, recurrent_dropout of 0.5
         # 3) a fully connected relu activation layer with 256 outputs,  4) a droupout layer 5) a final decision fully connected layer of
         # (which is the number of classes) with softmax activation
         model = tf.keras.Sequential([
             tf.keras.layers.Masking(mask_value=0.),
             tf.keras.layers.LSTM(512, dropout=0.5, recurrent_dropout=0.5),
             tf.keras.layers.Dense(256, activation='relu'),
             tf.keras.layers.Dropout(0.5),
             tf.keras.layers.Dense(len(LABELS), activation='softmax')
         ])

In [32]: model.compile(loss='categorical_crossentropy',
                       optimizer='rmsprop',
                       metrics=['accuracy', 'top_k_categorical_accuracy'])
```

setup a keras Sequential model




```
In [41]: # Setup the train_dataset and valid_dataset (validation/testing).
         # Here we setting up training batch sets of 16.

         train_dataset = tf.data.Dataset.from_generator(make_generator(train_list),
                          output_types=(tf.float32, tf.int16),
                          output_shapes=(tf.TensorShape([SEQUENCE_LENGTH, 1280]), tf.TensorShape([len(LABELS)])))
         train_dataset = train_dataset.batch(16,drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)


         valid_dataset = tf.data.Dataset.from_generator(make_generator(test_list),
                          output_types=(tf.float32, tf.int16),
                          output_shapes=(tf.TensorShape([SEQUENCE_LENGTH, 1280]), tf.TensorShape([len(LABELS)])))
         valid_dataset = valid_dataset.batch(16,drop_remainder=True).prefetch(tf.data.experimental.AUTOTUNE)

In [42]: print(train_dataset)

         <DatasetV1Adapter shapes: ((16, 40, 1280), (16, 37)), types: (tf.float32, tf.int16)>

In [44]: # fit the model with the training data (can increase the # of epochs & validation_steps if desired)
         model.fit(train_dataset, epochs=100, validation_data=valid_dataset, validation_steps=4)

         ical_accuracy: 0.36 - 116s 1s/step - loss: 3.2105 - acc: 0.0885 - top_k_categorical_accuracy: 0.36 - 117s 1s/step - loss: 3.2
         096 - acc: 0.0901 - top_k_categorical_accuracy: 0.36 - 118s 1s/step - loss: 3.2087 - acc: 0.0898 - top_k_categorical_accurac
         y: 0.36 - 119s 1s/step - loss: 3.2095 - acc: 0.0895 - top_k_categorical_accuracy: 0.36180/180 [============================
         =].2078 - acc: 0.0893 - top_k_categorical_accuracy: 0.36 - 122s 1s/step - loss: 3.2094 - acc: 0.0890 - top_k_categorical_accu
```

Set up the training dataset and validation dataset and fit the model with training data.



Save the model file in .H5 format.