Configuration Manual

MSc Research Project
Data Analytics

Rohan Narayan Koli
Student ID: 19224842

School of Computing
National College of Ireland

Supervisor: Prof. Majd Latifi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rohan Narayan Koli
Student ID:	19224842
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Prof. Majd Latifi
Submission Due Date:	16/08/2021
Project Title:	Classification of Speakers Age, Gender and Nationality
Word Count:	748
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rohan Narayan Koli
19224842

1. Introduction to configuration manual:

This configuration manual can be used to replicate the work done and get the desired results. It includes system configuration on which the project was run on, exploratory data analysis steps, model implementation and model evaluations. The code snippets are attached in the final section.

2. Pre-requisites and system configuration:

The tools and software used for this thesis research work can be installed on a laptop or a PC. The basic configuration list is given below:

Environment	Google Colab	Kaggle
RAM	16 GB	16 GB
Hard Disk	73 GB SSD	100 GB SSD
Processor	Intel Xeon 2.30GHz	Intel Xeon 2.30GHz
GPU	16 GB	13 GB

Getting started:

The basic toolset used in this research work for carrying out all the actions are listed below:

- Microsoft office tools
- Python 3.7.10
- Anaconda Spyder

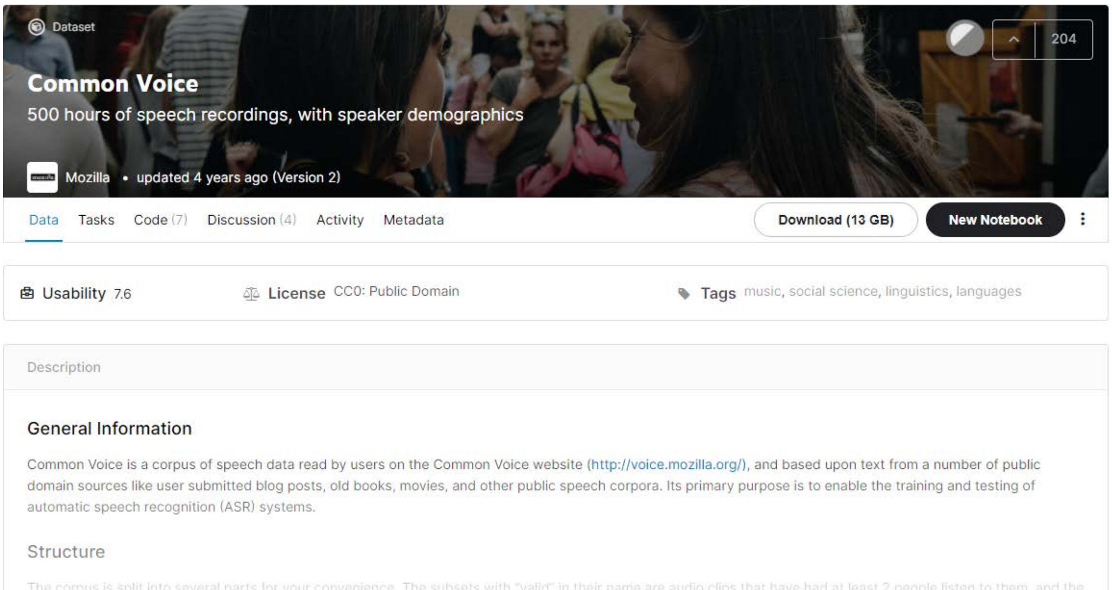
The Microsoft office tools like Microsoft Excel and Word have been used. Python as a language has been used for this research work and all the processes like data gathering, data cleaning, transformation and analysis has been done in python language. The software version for python used is 3.7.10 – 'https://www.python.org/downloads/'. The platform used for coding is Google Colab and Kaggle.

3. Database:

Two datasets are extracted from the following links and stored on Google Drive and Kaggle.

1. Mozilla Common Voice:

Link: https://www.kaggle.com/mozillaorg/common-voice?select=cv-valid-dev.csv



2. Speech Accent Dataset:

Link : https://www.kaggle.com/rtatman/speech-accent-archive

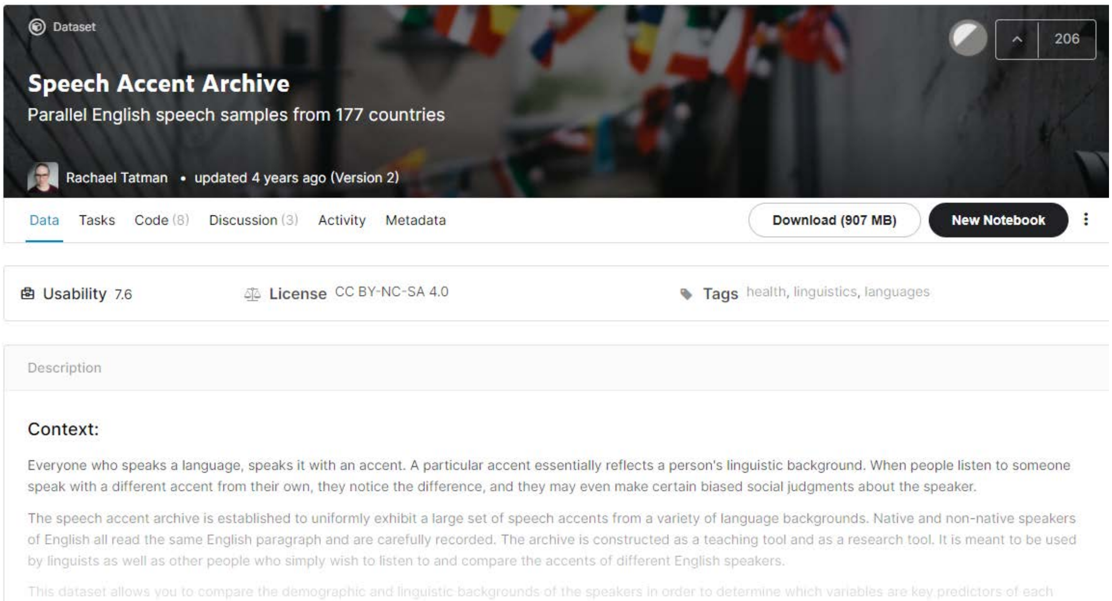


Figure 1: code snippet to convert the json file and clean the dataset.

3. Research design workflow and methodology:

We first process the audio signals, followed by pre-processing and extracting Mel Spectrograms. Next we transform the images into numpy arrays as an input vectors to the deep learning models.

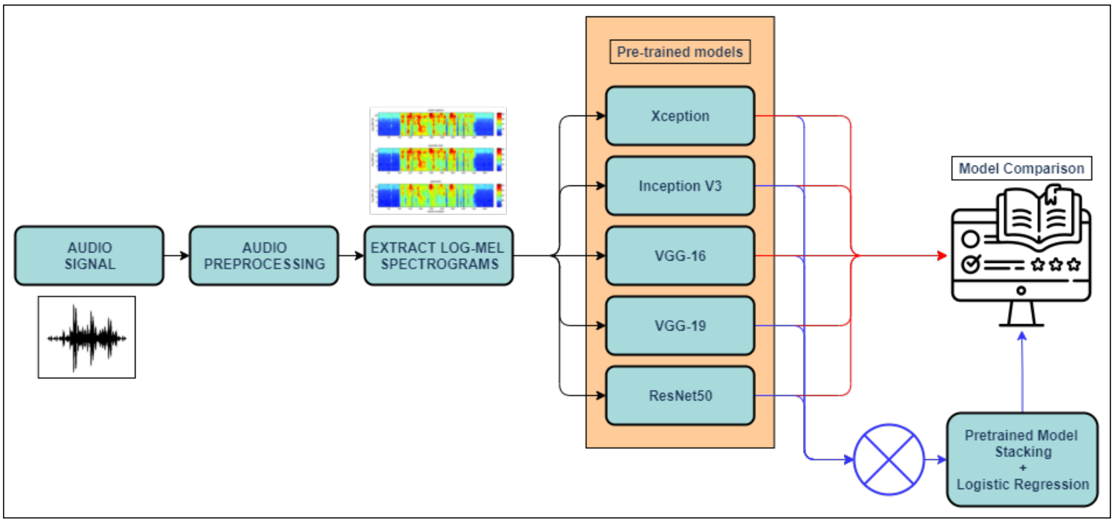


Figure 2: Design flowchart

4. Libraries used in code:

- Os : to make directories and manipulate files and directories
- numpy : Modelling, Data exploration
- pandas : Data modelling, visualization
- matplotlib: visualization
- seaborn: visualization
- librosa: Audio conversion
- sklearn.preprocessing MinMaxScaler: Normalize audio
- tqdm: get progress bar on loops
- sklearn.linear_model LogisticRegression: Model Stacking
- keras.preprocessing: built-in image preprocessor
- sklearn.metrics accuracy_score: evaluation metrics
- sklearn.metrics recall_score: evaluation metrics
- sklearn.metrics precision_score: evaluation metrics
- sklearn.metrics f1_score: evaluation metrics
- sklearn.metrics confusion_matrix: evaluation metrics
- sklearn.model_selection train_test_split: Splitting the dataset
- keras.applications resnet,vgg19,vgg16,xception,inception_v3: Model initializat
- keras.preprocessing.image ImageDataGenerator
- keras.models Sequential: Model layer
- keras.layers Flatten,BatchNormalization
- keras.layers Dense,Dropout: Model layers
- keras.optimizers Adam: Compiler
- plotly: Visualization

4. Data Preprocessing:

Generating images from audio samples.

```python
def process_data(file,target_dir):
    filename = voice_dir +  "/" + file
    y, s = librosa.load(filename, sr=16000)
    y_filt = librosa.effects.preemphasis(y)
    S_preemph = librosa.amplitude_to_db(np.abs(librosa.stft(y_filt)), ref=np.max)
    S_preemph = scaler.fit_transform(S_preemph)
    #librosa.display.specshow(S_preemph, y_axis='log', x_axis='time')
    plt.imshow(S_preemph.T,cmap='plasma')
    plt.axis("off")
    file = file.split("/")[1]
    address = target_dir+"/"+"{}.png".format(file)
    plt.savefig(address)
    plt.close()
```

```python
train_image = []
for i in tqdm(range(train.shape[0])):
    img = image.load_img('/content/drive/MyDrive/clean_random_images_10sec_cropped/'+train['filename'][i]+'.png',target_size=(224
    img = image.img_to_array(img)
    img = img/255
    train_image.append(img)
X = np.array(train_image)
```

```
100%|██████████| 2134/2134 [00:08<00:00, 250.70it/s]
```

5. Model Implementation:

Defining top layers for all the pre-trained model

```python
def dense_model(base_model,num_classes):
    model = Sequential()
    model.add(base_model)
    model.add(Flatten())
    model.add(BatchNormalization())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(num_classes, activation='softmax'))

    return model
```

Defining All the models

```python
def define_models(classes):
    num_class = classes
    #Xception
    model_xcep = Xception(include_top=False, weights="imagenet",input_shape =inputShape)
    model_xception = dense_model(model_xcep,num_class)
    model_xception.layers[0].trainable = False

    # InceptionV3
    model_incep = InceptionV3(include_top=False, weights="imagenet",input_shape =inputShape)
    model_inception = dense_model(model_incep,num_class)
    model_inception.layers[0].trainable = False

    #VGG16
    model_1 = VGG16(include_top=False, weights="imagenet",input_shape =inputShape)
    model_vgg1 = dense_model(model_1,num_class)
    model_vgg1.layers[0].trainable = False

    #VGG 19
    model_2 = VGG19(include_top=False, weights="imagenet",input_shape =inputShape)
    model_vgg2 = dense_model(model_2,num_class)
    model_vgg2.layers[0].trainable = False

    #ResNet50
    model_res = ResNet50(include_top=False, weights="imagenet",input_shape =inputShape)
    model_resnet = dense_model(model_res,num_class)
    model_resnet.layers[0].trainable = False

    return [model_xception,model_inception,model_vgg1,model_vgg2,model_resnet]
```

6. Stacked model Implementation:

```
def stacking_predictions(models,data):
    # array to store values
    stackValues = None
    for model in models:
        # making predictions for each model
        y_pred = model.predict(data)
        # stack predictions into [rows, members, probabilities]
        if stackValues is None:
            stackValues = y_pred
        else:
            stackValues = np.dstack((stackValues,y_pred))
    # flatten predictions to [rows, members x probabilities]
    stackValues = stackValues.reshape((stackValues.shape[0], stackValues.shape[1]*stackValues.shape[2]))
    return stackValues
```

```
def fit_models(models,data,labels):
    # stacked data with ensemble
    stackedValues = stacking_predictions(models,data)
    log_reg = LogisticRegression()
    labels = np.argmax((labels.values),axis=1)
    log_reg.fit(stackedValues,labels)
    return log_reg
```

```
def stacked_prediction(members, model, inputX):
    # create dataset using ensemble
    stackedX = stacking_predictions(members, inputX)
    # make a prediction
    yhat = model.predict(stackedX)
    return yhat
```

Model Training:

Gender prediction

```
label = pd.get_dummies(train['sex'])
X_train, X_test, y_train, y_test = train_test_split(X, label, random_state=42, test_size=0.2)
```

```
model_xception,model_inception,model_vgg1,model_vgg2,model_resnet = define_models(2)
```

Exception Model

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model_xception.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["accuracy",recall_score,precision_score,f1_score])
# train the network
print("[INFO] training network...")
history_exception = model_xception.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

```
[INFO] training network...
Epoch 1/10
54/54 [==============================] - 44s 238ms/step - loss: 0.9013 - accuracy: 0.5652 - recall_score: 0.5653 - precision_sc
ore: 0.5653 - f1_score: 0.5653 - val_loss: 0.5412 - val_accuracy: 0.7728 - val_recall_score: 0.7750 - val_precision_score: 0.77
50 - val_f1_score: 0.7750
Epoch 2/10
54/54 [==============================] - 9s 160ms/step - loss: 0.6606 - accuracy: 0.6347 - recall_score: 0.6347 - precision_sco
re: 0.6347 - f1_score: 0.6347 - val_loss: 0.4917 - val_accuracy: 0.7564 - val_recall_score: 0.7593 - val_precision_score: 0.759
3 - val_f1_score: 0.7593
Epoch 3/10
```

AGE Group Prediction

```
label = pd.get_dummies(train['age'])
X_train, X_test, y_train, y_test = train_test_split(X, label, random_state=42, test_size=0.2)
```

```
model_xception,model_inception,model_vgg1,model_vgg2,model_resnet = define_models(y_train.shape[1])
```

Xception

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model_xception.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["accuracy",recall_score,precision_score,f1_score])
# train the network
print("[INFO] training network...")
history_xception = model_xception.fit(X_train, y_train.values, epochs=10, validation_data=(X_test, y_test.values))
```

```
[INFO] training network...
Epoch 1/10
54/54 [==============================] - 22s 238ms/step - loss: 1.5755 - accuracy: 0.3525 - recall_score: 0.2767 - precision_sc
ore: 0.3510 - f1_score: 0.3090 - val_loss: 1.0488 - val_accuracy: 0.4192 - val_recall_score: 0.0179 - val_precision_score: 0.25
00 - val_f1_score: 0.0330
Epoch 2/10
54/54 [==============================] - 10s 187ms/step - loss: 1.4316 - accuracy: 0.3479 - recall_score: 0.2741 - precision_sc
ore: 0.3527 - f1_score: 0.3079 - val_loss: 1.0127 - val_accuracy: 0.4239 - val_recall_score: 0.0467 - val_precision_score: 0.33
87 - val_f1_score: 0.0808
Epoch 3/10
```

Accent Prediction

```
label = pd.get_dummies(train['Continent'])
X_train, X_test, y_train, y_test = train_test_split(X, label, random_state=42, test_size=0.2)
```

```
model_xception,model_inception,model_vgg1,model_vgg2,model_resnet = define_models(y_train.shape[1])
```

Xception

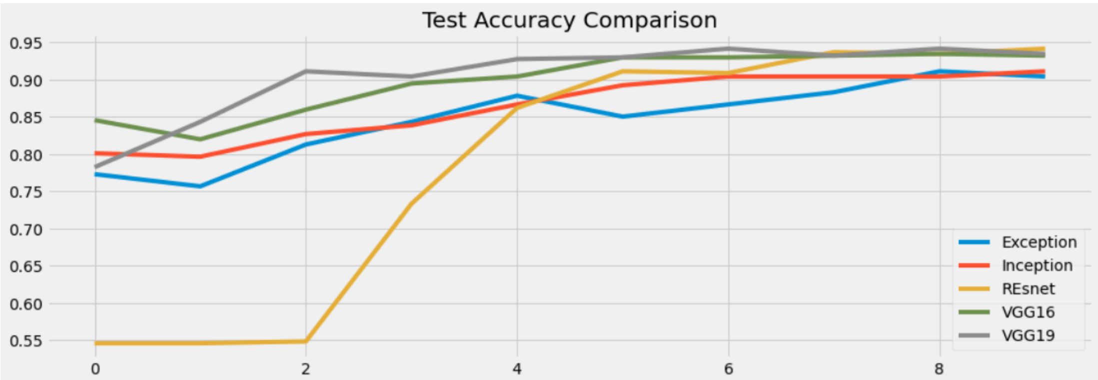
```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model_xception.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["accuracy",recall_score,precision_score,f1_score])
# train the network
print("[INFO] training network...")

history_xception = model_xception.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

```
[INFO] training network...
Epoch 1/10
54/54 [==============================] - 23s 235ms/step - loss: 2.3890 - accuracy: 0.1675 - recall_score: 0.0721 - precision_sc
ore: 0.2030 - f1_score: 0.1056 - val_loss: 1.7708 - val_accuracy: 0.2459 - val_recall_score: 0.0000e+00 - val_precision_score:
0.0000e+00 - val_f1_score: 0.0000e+00
Epoch 2/10
54/54 [==============================] - 9s 172ms/step - loss: 2.1789 - accuracy: 0.1870 - recall_score: 0.0691 - precision_sco
re: 0.2313 - f1_score: 0.1057 - val_loss: 1.7022 - val_accuracy: 0.3091 - val_recall_score: 0.0000e+00 - val_precision_score:
0.0000e+00 - val_f1_score: 0.0000e+00
Epoch 3/10
54/54 [==============================] - 9s 175ms/step - loss: 1.9669 - accuracy: 0.2106 - recall_score: 0.0560 - precision_sco
```

7. Model Evaluation:

```
plt.figure(figsize=(15,5))
plt.title("Test Accuracy Comparison")
plt.plot(history_exception.history["val_accuracy"],label = "Exception")
plt.plot(history_inception.history["val_accuracy"],label = "Inception")
plt.plot(history_resnet.history["val_accuracy"],label = "REsnet")
plt.plot(history_vgg1.history["val_accuracy"],label = "VGG16")
plt.plot(history_vgg2.history["val_accuracy"],label = "VGG19")
plt.legend()
plt.show()
```



Finally, we apply the same steps using the Speech Accent Dataset.