[2020]

CAB420: Machine Learning

ASSIGNMENT 1

MALVIKA AMARNATH (N9519661)

QUEENSLAND UNIVERSITY OF TECHNOLOGY |

Contents

Question 1:

Q1. How Data is Handled (Data Cleaning or Removal and splitting of Data)

Data consists of non-predictive features such as State, County, Community, Communityname and fold. Data Consists of 123 columns and 1994 rows.

- We are dropping the non-predictive features because it does not add any value to the dependent variable or Target Variable.
- We are dropping the columns which contain more than 80% of null values. Because it will reduce the performance and it will also take more time and space which interns reduce the performance of the model
- ViolentCrimesPerPop is the target variable which consists of total number of violent crimes per 100k population.
- Data is divided into

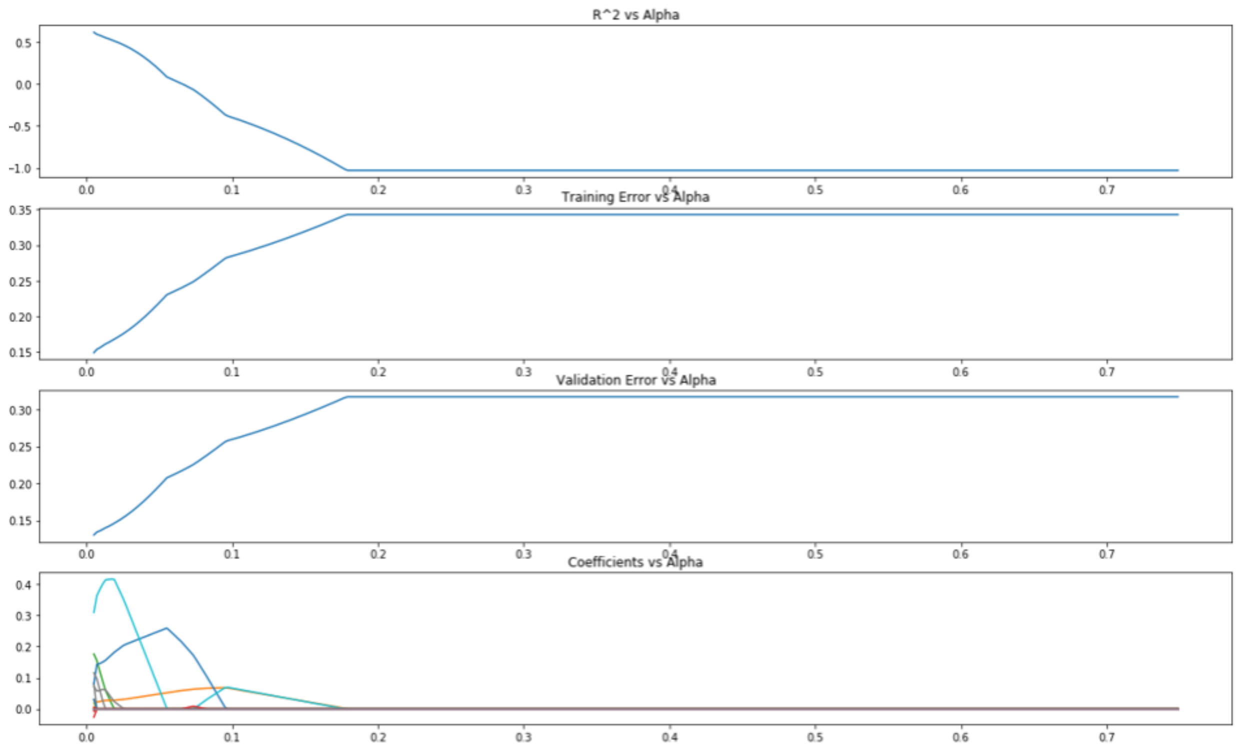
Training	60%
Validation	20%
Testing	20%

Q2. Details of the three trained models, including details such as values for λ for the LASSO and Ridge models, and a brief discussion of how these were selected.

Lasso:

We took the range of λ values from 0.005 to 0.75 with an increment of 0.002. We find the best λ values from the above range.

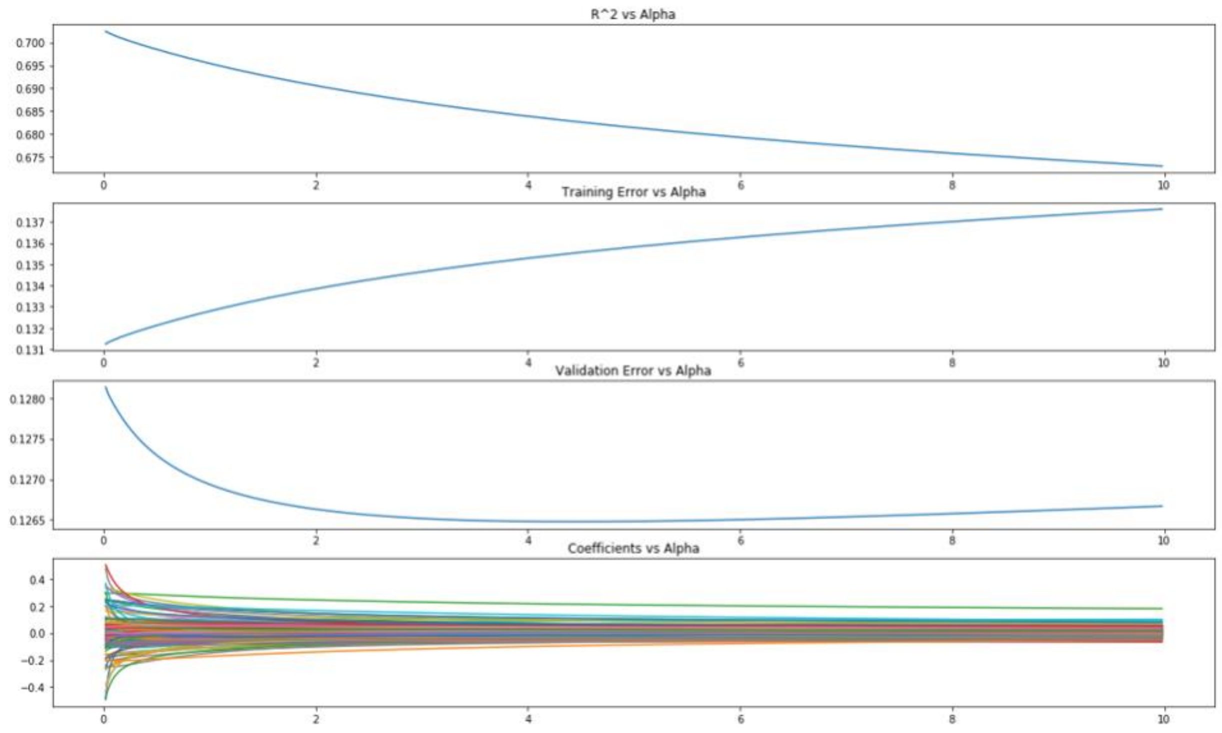
- R^2 value goes down as λ increase and it stops reducing at the value 0.1 to 0.2
- Training Error Goes up as λ increases. Training error stopped when λ value reached the value between 0.1 and 0.2
- The best Lasso λ value is 0.005. But the main disadvantage of using λ is it shrinks the attribute or features. But it is suitable for variable selection.
- In the last figure, X axis represents the λ value and Y axis represents the coefficient associated with each attributes or features.



Ridge:

We took the range of λ values from 0.02 to 10 with an increment of 0.02. We find the best λ values from the above range.

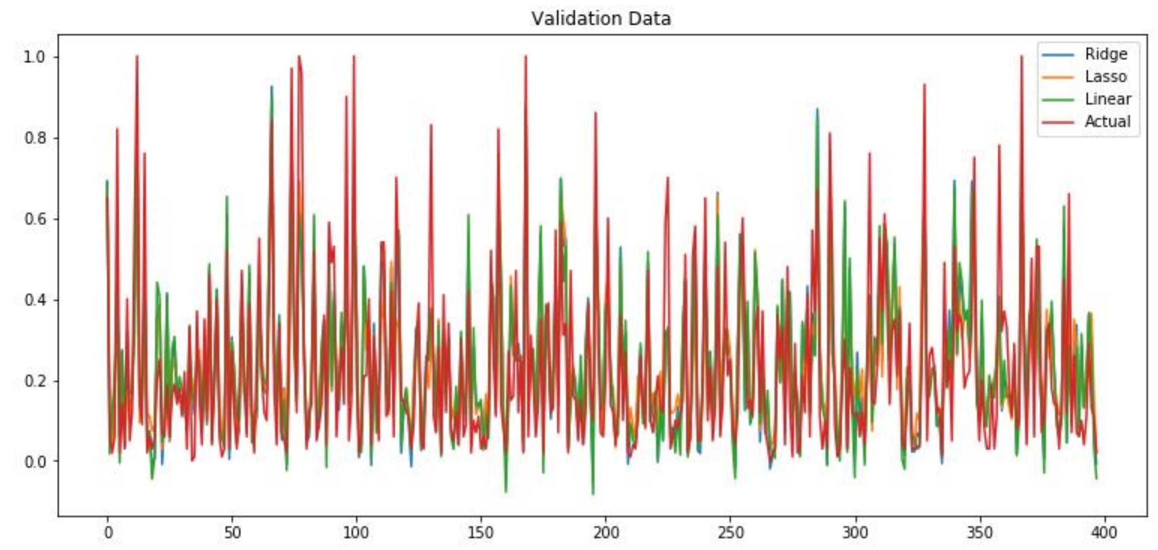
- R^2 goes down as λ increases.
- Training Error Goes up with the increase in λ value.
- Validation Error Goes down like an elbow shape with sharp bend at the λ value of 2.
- The best Lasso Alpha is 4.4399
- In the last figure, X-axis represents the λ value and Y-axis represents the coefficient associated with each attributes or features.

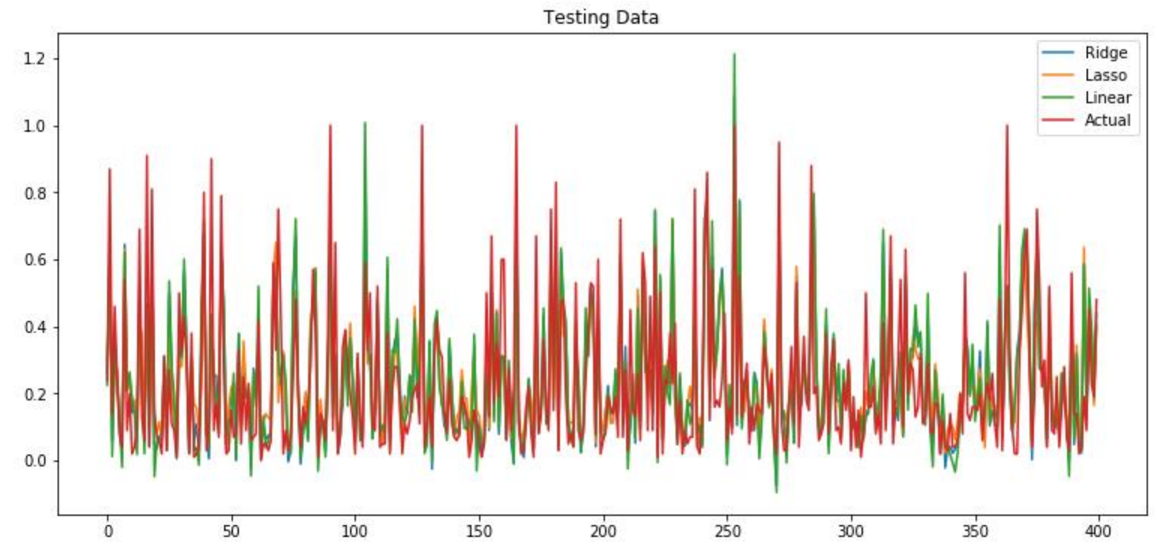


Q3. An evaluation comparing the three models, considering model accuracy and model validity***.***

By looking at the RMSE's on the test set, we can say that the Ridge model performed better than Linear and Lasso Model. Here, the Lasso model didn't perform well. The main reason is the Lasso model created sparsity in the above dataset and removed many attributes. Ridge helped in the generalization of the test dataset. Lasso created a sparse model

Linear Model	Test RMSE: 0.13663131077015342
Lasso Model,	Test RMSE: 0.14032160148380318
Ridge Model,	Test RMSE: 0.13337810483643836


Validation Data
Ridge
Lasso
Linear
Actual
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400


Testing Data
Ridge
Lasso
Linear
Actual
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400

Question 2: Classification

Q1. Details on how the data was split into training, validation and testing sets.

Data is divided into Training and Testing. In which Training sets can be used to feed the model and Testing Sets are used to evaluate the model.

Training	37%
Testing	63%

Q2. Details of hyper-parameter selection, including a justification for the approach taken and any intermediate results that led to the final models. hyper-parameter selection

Hyper-parameters help to generalize the Models for real-world problems. It will help to increase the accuracy of the model. Users need to hard code or enter the hyperparameters. In this Multi-class classification, we are using mainly K-NN, SVMs.

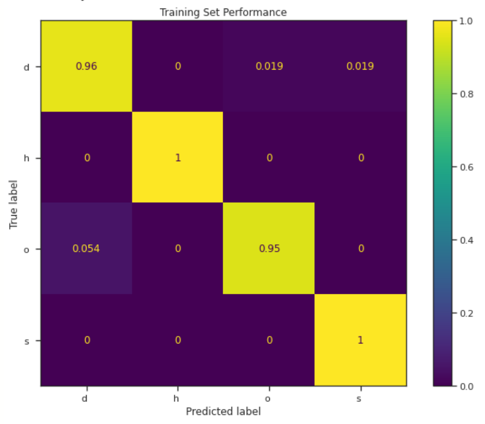
1. **K-NN:**

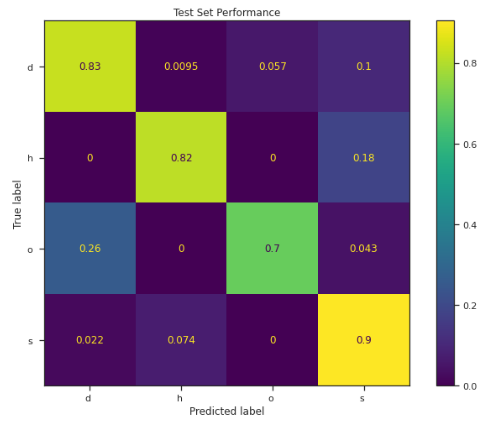
k-nearest neighbour classifies the data based on its neighbours. If the majority of the neighbour vote for 'yes' then it classifies the unknown dataset to 'yes'

Several neighbours are considered as hyperparameter for K-NN model. Because there is no default value or optimize value to select the exact number of neighbours. We can also use the elbow method to select the optimal number of neighbours. Below Graphs represents the number of neighbours and their accuracy

a. Number of neighbours = 3 and the Accuracy is 84%

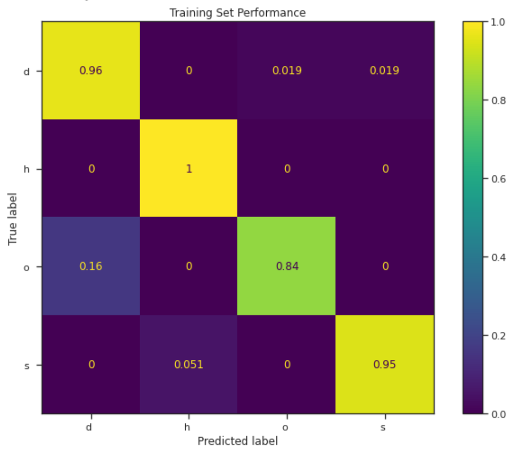
```
Test Accuracy: 0.84
```

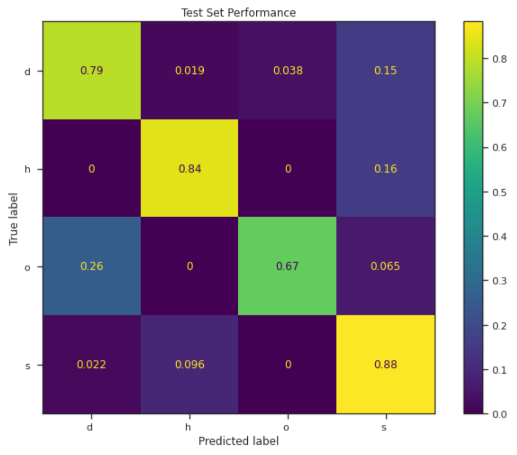




b. Number of neighbours = 10 and the Accuracy is 81%

```
Test Accuracy: 0.8184615384615385
```

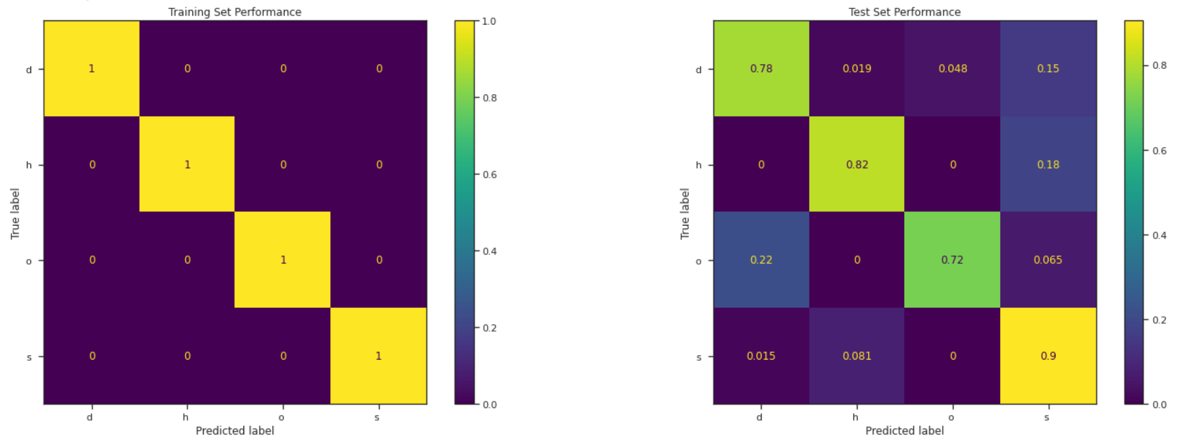




c. Number of neighbours = 10, weights is distance and the Accuracy is 82%

Here, we are using another hyperparameter such as ‘distance’. This means closer datasets have more weightage than other datapoints. Which helps to increase the accuracy from 81% to 82%.

```
Test Accuracy: 0.8276923076923077
```



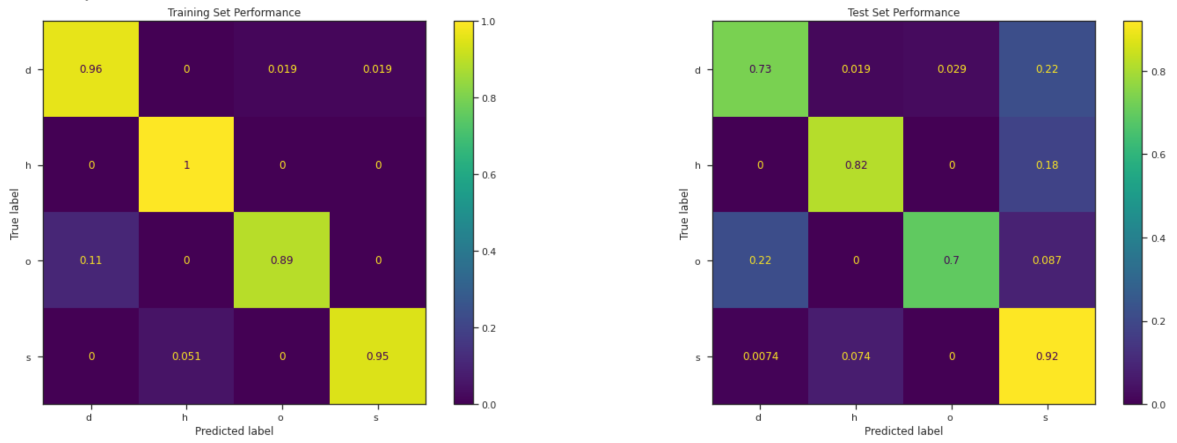
2. Multi-Class SVMs

Support Vector Machine helps to find the hyperplane in an N-dimensional space that distinctly classifies the data points. Here, Hyperplane acts as a decision boundary.

a. **Default** SVM without any parameter and the accuracy is 81%

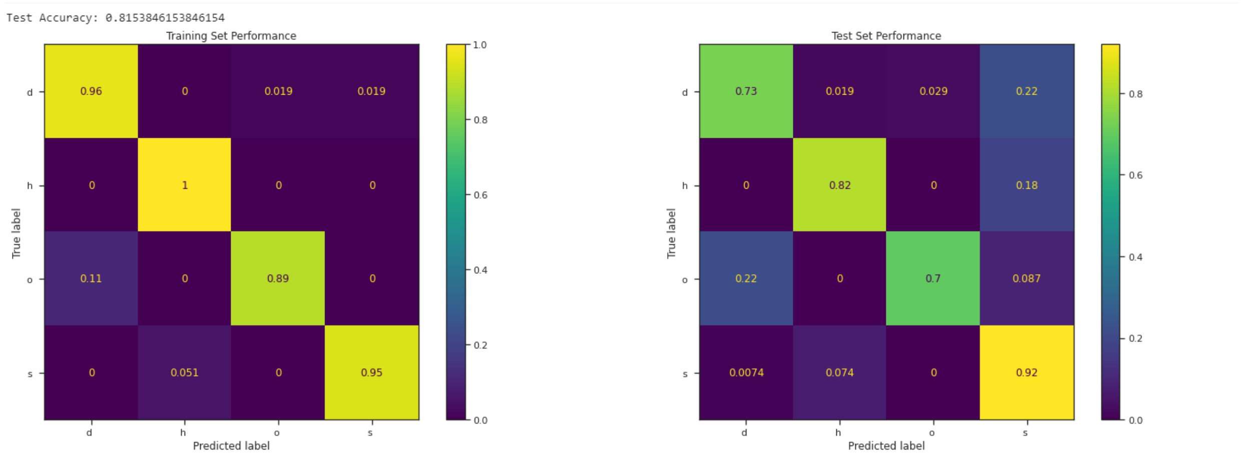
We are not using any hyperparameter here.

```
Test Accuracy: 0.8153846153846154
```



b. SVM with nu value is 0.15 and the accuracy is 81%

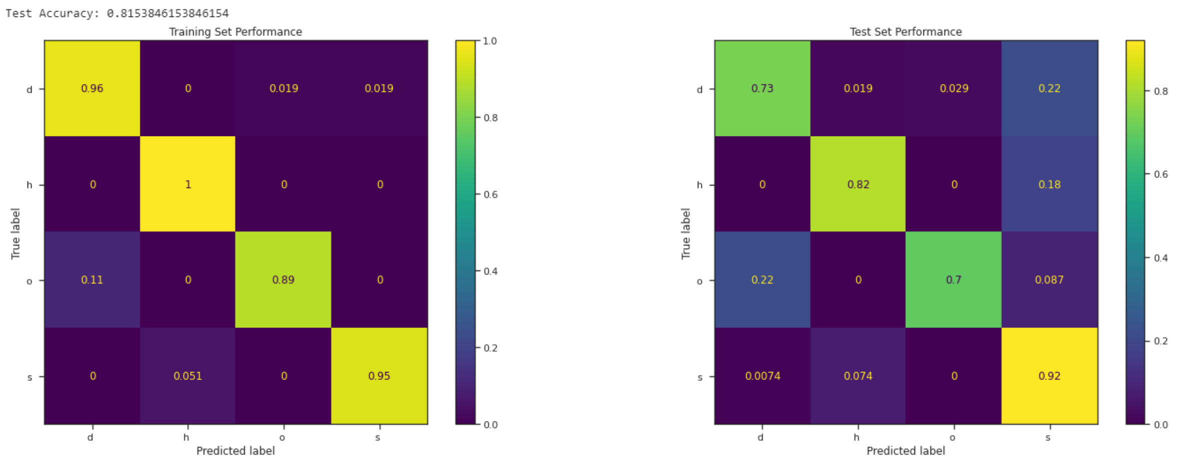
nuSVC is similar to SVM but it uses the parameter to control the number of support vectors. Where nu value must range between 0 to 1 which represents the training errors in a upper bound of fraction and Support Vectors on the lower bound of the fractions.



```
Test Accuracy: 0.8153846153846154
```

c. SVM with linear kernel and the accuracy is 83%

Here we are using another hyperparameter such as ‘Kernel’. There are 5 different types of kernel and they are ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed. We selected ‘linear’ and the accuracy increased from 81% to 83%.



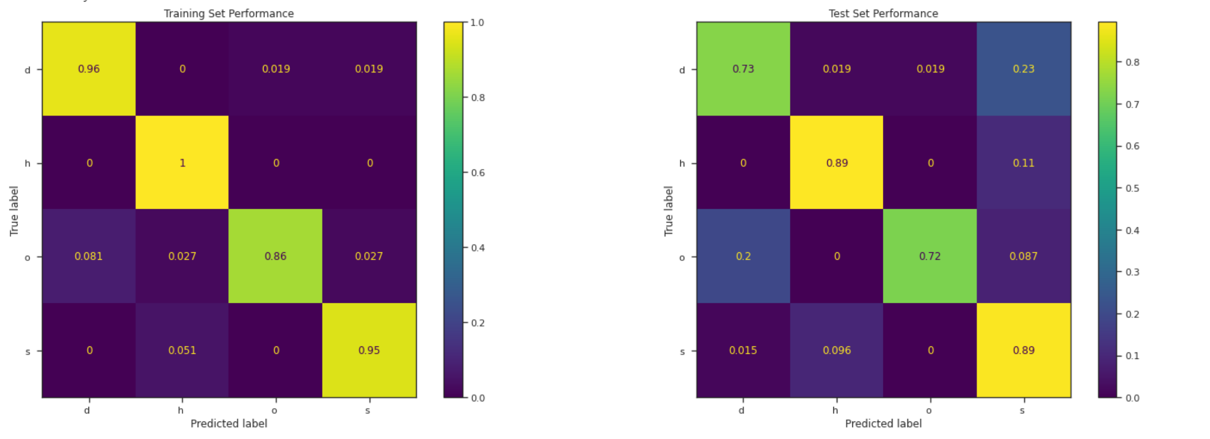
```
Test Accuracy: 0.8153846153846154
```

3. Ensemble of Binary Classifier (One Vs All).

We are creating One Vs All model by using Support Vector Machine model. One vs All will help to train one classifier per class. Which take one target class and separate it with all the other class. For example it will take ‘Sugi Forest’ as one class and rest as other class.

Here, we are using class_weight as the hyperparameter. Balanced class class_weight helps to adjust the weights inversely proportional to class frequencies. We got the accuracy of 81% which is less than K-NN model.

```
Test Accuracy: 0.8153846153846154
```



Q3. An evaluation and comparison of the final two models, including a discussion exploring any difference in performance between the models.

Model	**Accuracy**
K-NN model (neighbours = 3)	0.84
SVM model (C = numpy.inf, kernel='linear')	0.8369230769230769
One Vs All (Balanced)	0.8153846153846154

By Comparing all the above model, K-NN model with the number of neighbours 3 performed better than the rest of the model. Here, K-NN and SVM accuracy is roughly the same. We can also play around by changing the kernel type from linear to other types and check the accuracy. 'Free Lunch Theorem' suggest that no model that works best for every problem. Hence, We need to play around with different hyperparameter and select the best one.

Question 3:

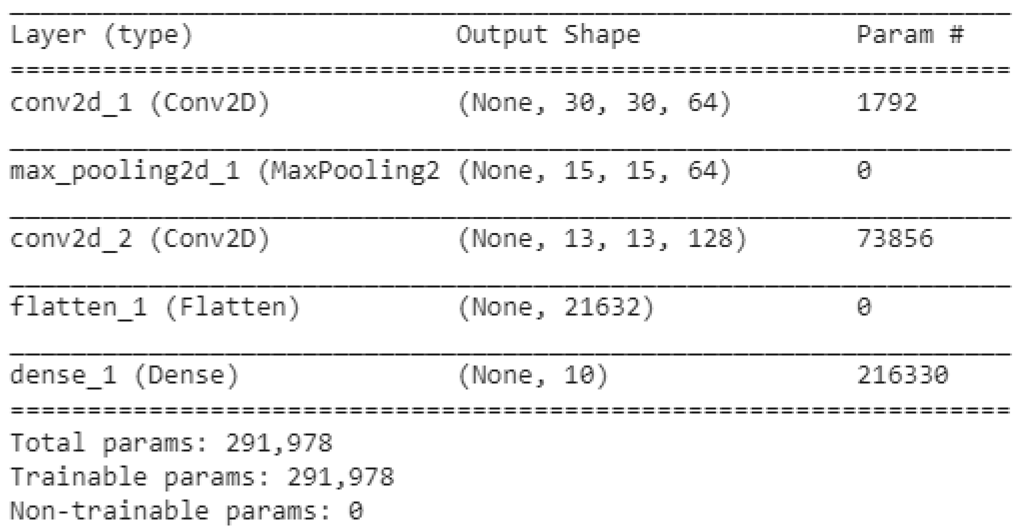
Q1: Ensure that all choices (e.g. network design, type of augmentation) are explained and justified in your response.

CNN with No Data Augmentation:

Network Design:

I implemented a small Network which consists of 2 Convolution layer, 1 Max pooling layer and 1 Flatten layer. The Network is fully connected (Dense). SVHN dataset consists of digits from 1 to 10 hence, the output layer is 10.
Activation function is ‘relu’ which helps to find non-linear relationships.

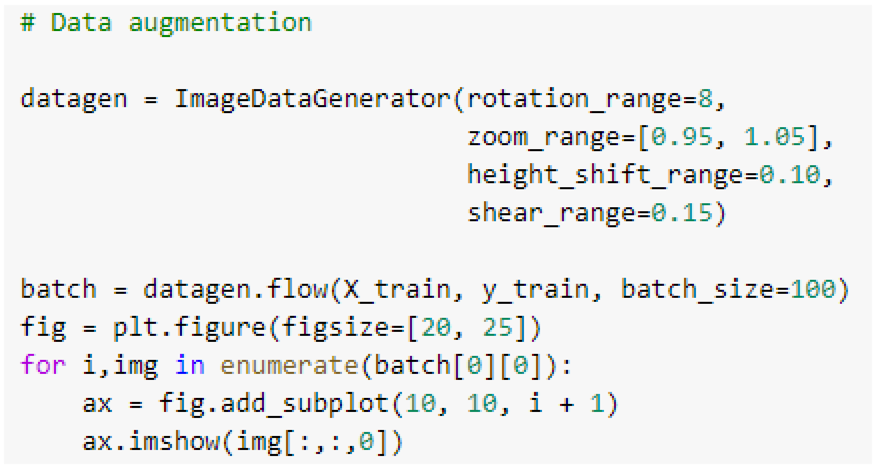
In the Output Layer, I used the ‘SoftMax’ function, which helps to select the digit with more probabilistic value.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 64)        1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 128)       73856
_________________________________________________________________
flatten_1 (Flatten)          (None, 21632)             0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                216330
=================================================================
Total params: 291,978
Trainable params: 291,978
Non-trainable params: 0
```

CNN with Data Augmentation:

We are using the Data Augmentation to increase the training samples. Here, the number of training data is very less. Data Augmentation will help to increase the number of training data. Below code helps to change the shape of the image by twisting, turning and rotating it.

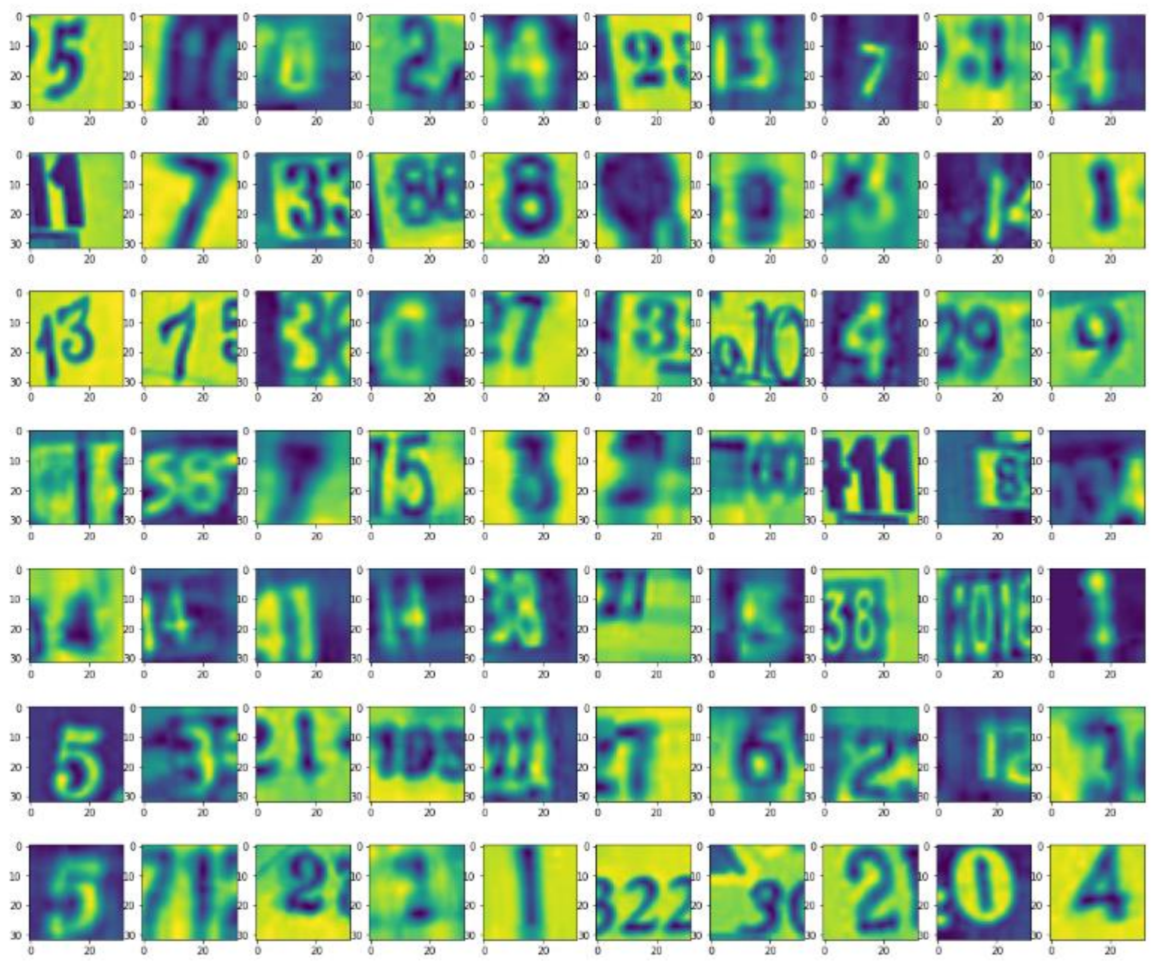


```
# Data augmentation

datagen = ImageDataGenerator(rotation_range=8,
                             zoom_range=[0.95, 1.05],
                             height_shift_range=0.10,
                             shear_range=0.15)

batch = datagen.flow(X_train, y_train, batch_size=100)
fig = plt.figure(figsize=[20, 25])
for i,img in enumerate(batch[0][0]):
    ax = fig.add_subplot(10, 10, i + 1)
    ax.imshow(img[:,:,0])
```

Rotation range indicates the range of image rotation which I mentioned as 8. Zoom_range defines the rage of random zoom. Where the first parameter indicates the lower range and the second parameter indicates the upper range of random zoom [lower , upper]. Shear_range defines the counter-clockwise rotation of the images. Few Images from the below dataset doesn't present in the real dataset. We have augmented it in to increase the training dataset.



Existing Model: VGG16

We are using the Existing model such as VGG16 to train our SVHN dataset by using Convolutional Neural Network. VGG16 was first proposed by Simonyan and A. Zisserman. We are using VGG16 to increase accuracy. We can see that our Training error is less and the testing error is more. Which indicates the overfitting. VGG16 may help to reduce the overfitting by adding a more convolutional layer.

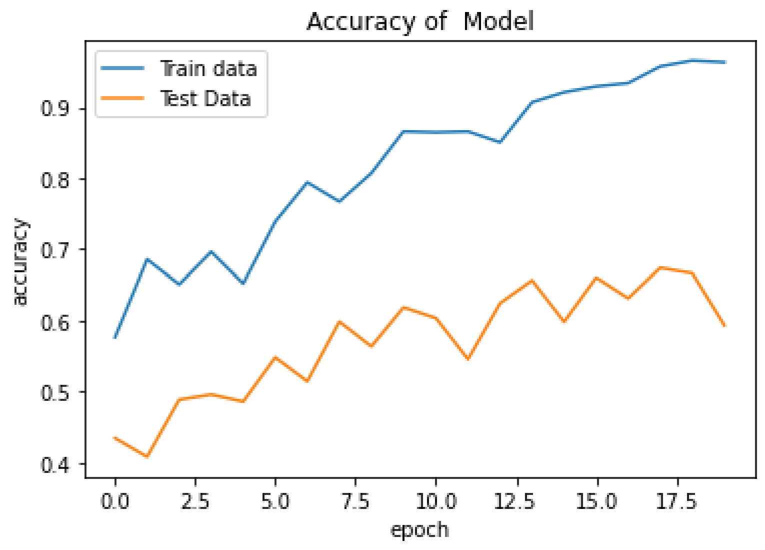
Question 2: Consider computational constraints. You do not need to train the most complex model possible. It is acceptable to use a simpler architecture due to computational constraints, though e_orts should still be made to achieve a good *level of performance and detailsregarding these choices and the tradeo_ between computational load and performance should be stated in the response.*

Question 3: Include all relevant _gures and/or tables to support the response.

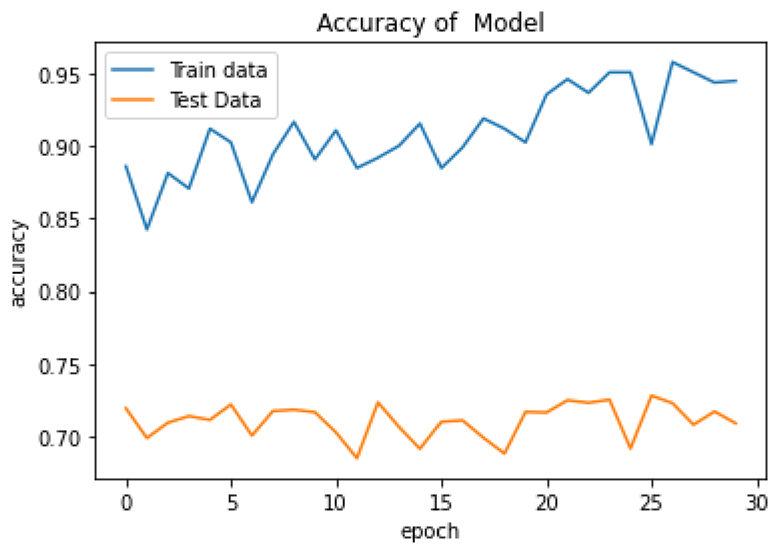
The Dataset is divided into 1000 training samples and 10,000 testing samples which is inappropriate.

Training	10%
Testing	90%

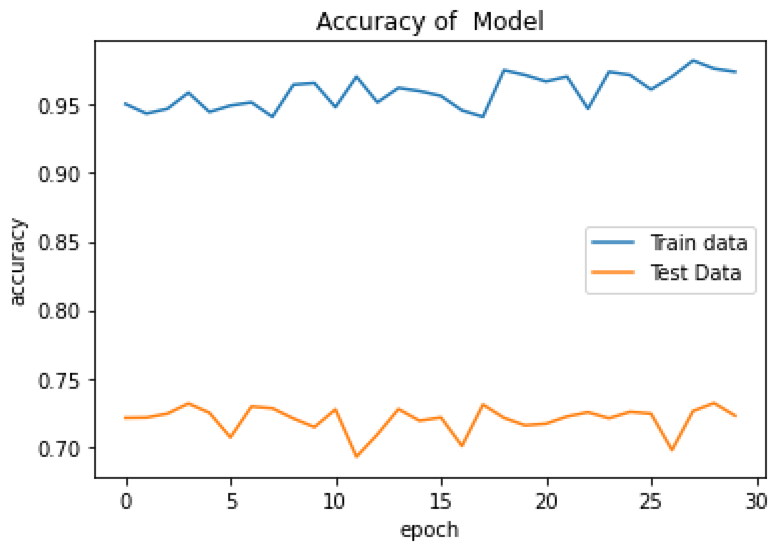
Which will leads to overfitting of the data. Overfitting means when the training accuracy is more and testing accuracy is less. Because the model fails to generalise the given dataset. The below table indicates the Training and Testing Accuracy of a model with no Augmentation.



Data Augmentation helps to increase the number of training samples which interns increases the accuracy of the model. The training accuracy of the mode increased from 59% to 69%. Below table shows the improvisation of Testing Accuracy.



Still, there is a huge gap between the training and testing Accuracy. We are using the VGG16 pre-trained model to generalize the model by reducing the Overfitting. The below diagram shows the performance of VGG16 model with Data Augmentation.



The testing accuracy is not improved much. The testing accuracy slightly increased from 69% to 71%. Increasing the number of convolutions in VGG16 did not add much value to the accuracy. But, there are so many other techniques there to minimize the overfitting. They are

1. Increasing the Training dataset: The main problem associated without dataset is a number of the training dataset. Which is very less. Hence, our

model fails to generalize the testing dataset. If we increase the training dataset then we can improvise the performance of the model.

2. Regularization: We can also improve the performance by adding Regularization (Dropouts, L1/L2 Regularization).

Question 4:

Q1: Develop a method to determine that age of a person in an image. There are multiple

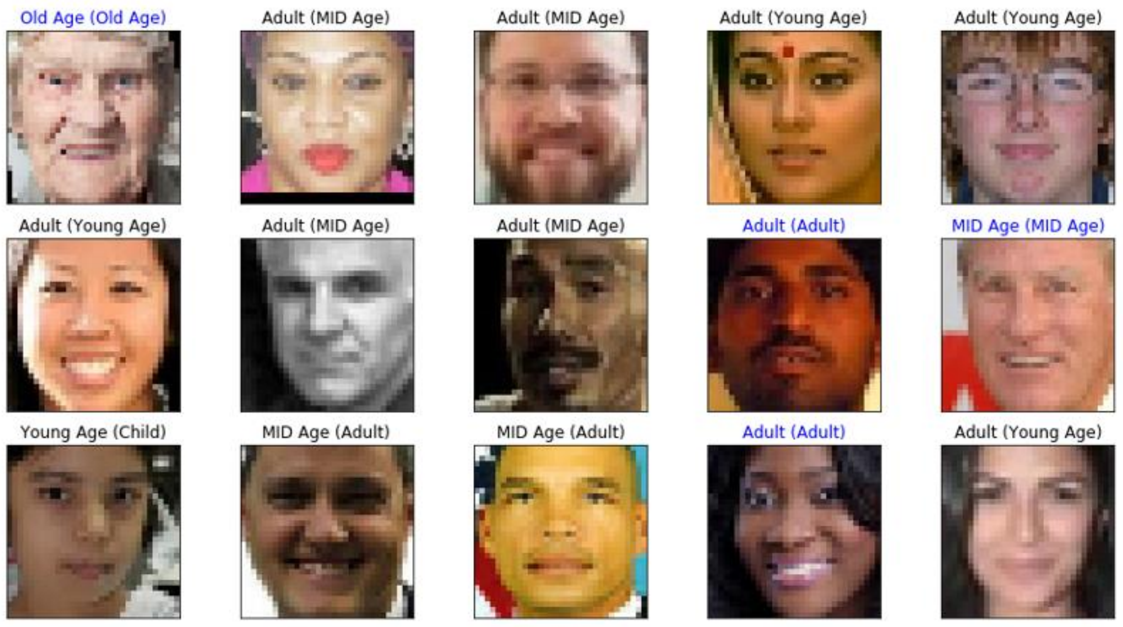
ways to approach this, including:

a. As a regression task, where you regress from the image (or features from an image) to the age.
b. As a classification task, where an image is classified into its age. Note that in particular for a classification approach, you may wish to classify the age into ranges (such as decades) rather than having one class for each age.

I created the classification to determine the age which of a person who is present is a Dataset. Classification involves.

Classification	**Age**
Child	Less than 14
Young Age	14 to 25
Adult	25 to 40
Mid Age	40 to 60
Old Age	More than 60

The below diagram represents the age classification of the person which is present in the dataset.



We can see that our model is ‘underfitting’. The main reason is the division of Data. We need to improvise the model by adding drop out function. Our model is too simple. We also need to normalize the given data to improve the accuracy and overcome the overfitting. We need to complicate the model by adding more layers to increase accuracy.

We can also overcome the overfitting if we increase the number of epochs from 5 to 50. It may add value to the performance of the model.

Question 2: Create a hold-out evaluation protocol, where 70% of the data is used for training, 15% is used for validation, and 15% is used for testing. Evaluate the model using thi protocol.

The Data is divided into 70% Training, 15% Validation and 15% Testing.

Question 3: **Create a cross-fold evaluation protocol based on the race annotation, i.e. your shouldcreate for folds of the data with each fold containing all instances of one race. The evaluation should then train the model on 4 of the folds (i.e. 4 races), and the test on the unseen race. This should be repeated such that all folds are the test set in turn.**

The Data Consists of 4 race and they are

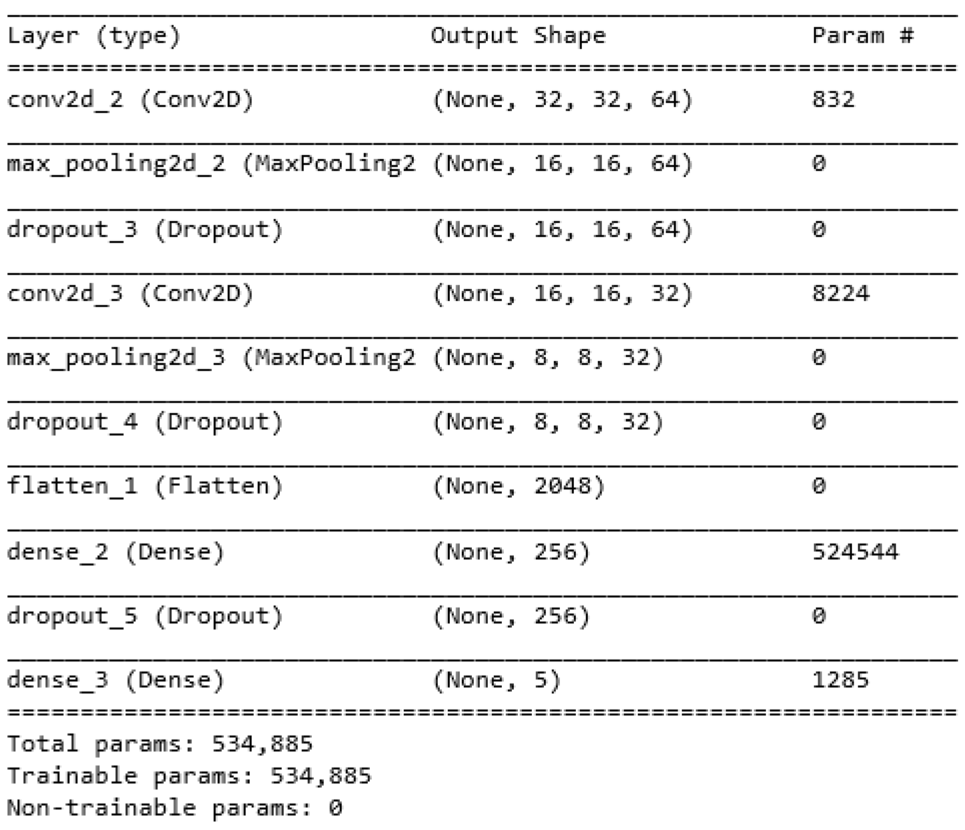
White	0
Black	1
Asian	2
Indian	3
Others	4

a. Briey explain the method you have chosen to use, and provide a brief justification your choice.

I'm using Convolutional Neural Network. I created a Simple Network to determine race and age.

Network Design (Age Prediction):

I created a Simple CNN design to predict the Age. I divided the dataset into 70% training and Remaining Validation(15%) and Testing (15%).



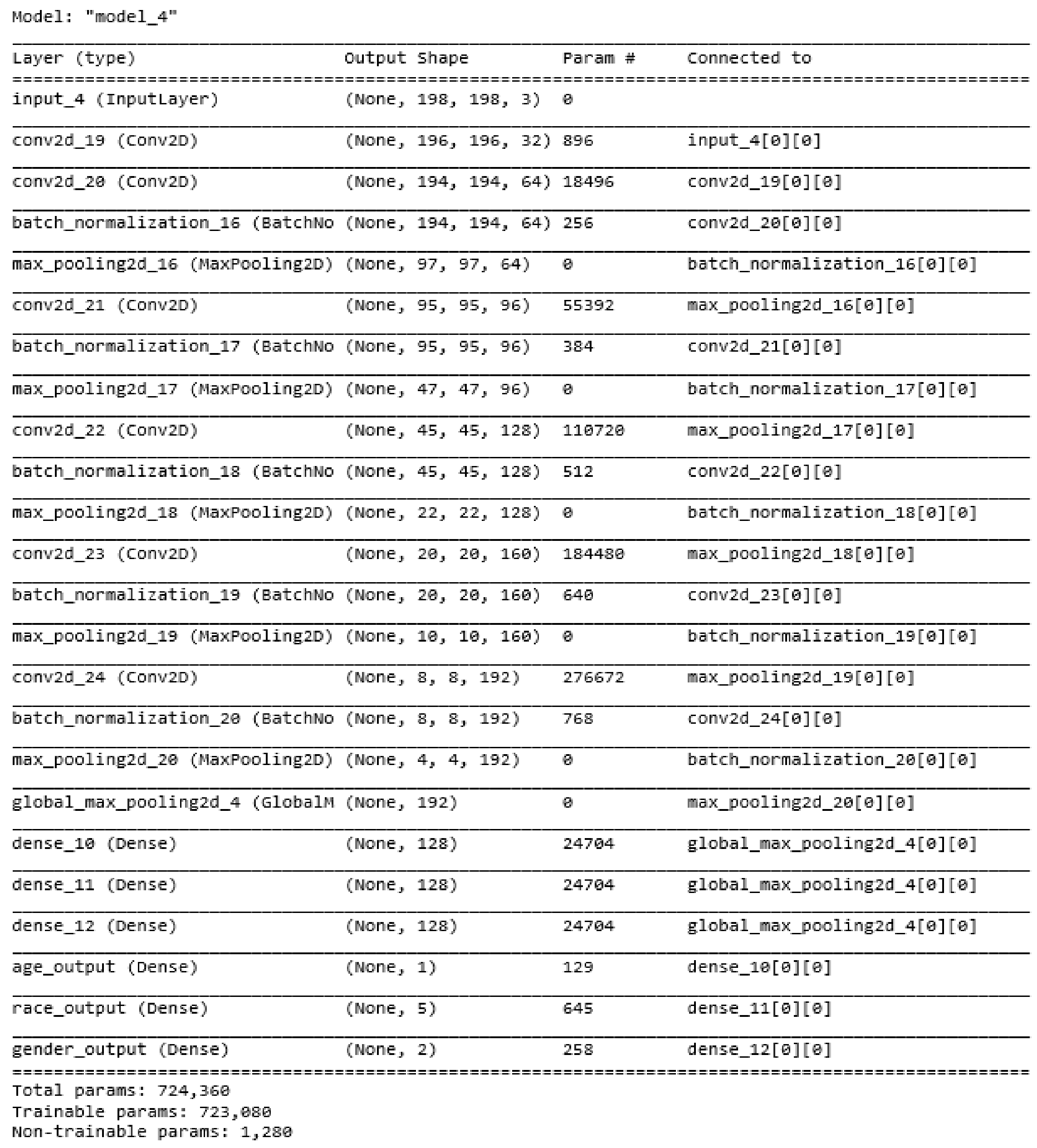
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 32, 32, 64)        832
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 32)        8224
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 8, 8, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               524544
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 1285
=================================================================
Total params: 534,885
Trainable params: 534,885
Non-trainable params: 0
```

Model Performance:

Training Accuracy: 58%
Test Accuracy: 59%

Network Design (Race Prediction):

I Created a little complicated model and I got better Accuracy. I implemented Batch Normalization, Dropouts to reduce the Overfitting. I divided the dataset into 70% training and Remaining Validation(15%) and Testing (15%).



```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 198, 198, 3)  0
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 196, 196, 32) 896         input_4[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 194, 194, 64) 18496       conv2d_19[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, 194, 194, 64) 256         conv2d_20[0][0]
__________________________________________________________________________________________________
max_pooling2d_16 (MaxPooling2D) (None, 97, 97, 64)   0           batch_normalization_16[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 95, 95, 96)   55392       max_pooling2d_16[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 95, 95, 96)   384         conv2d_21[0][0]
__________________________________________________________________________________________________
max_pooling2d_17 (MaxPooling2D) (None, 47, 47, 96)   0           batch_normalization_17[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 45, 45, 128)  110720      max_pooling2d_17[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 45, 45, 128)  512         conv2d_22[0][0]
__________________________________________________________________________________________________
max_pooling2d_18 (MaxPooling2D) (None, 22, 22, 128)  0           batch_normalization_18[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 20, 20, 160)  184480      max_pooling2d_18[0][0]
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, 20, 20, 160)  640         conv2d_23[0][0]
__________________________________________________________________________________________________
max_pooling2d_19 (MaxPooling2D) (None, 10, 10, 160)  0           batch_normalization_19[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 8, 8, 192)    276672      max_pooling2d_19[0][0]
__________________________________________________________________________________________________
batch_normalization_20 (BatchNo (None, 8, 8, 192)    768         conv2d_24[0][0]
__________________________________________________________________________________________________
max_pooling2d_20 (MaxPooling2D) (None, 4, 4, 192)    0           batch_normalization_20[0][0]
__________________________________________________________________________________________________
global_max_pooling2d_4 (GlobalM (None, 192)          0           max_pooling2d_20[0][0]
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 128)          24704       global_max_pooling2d_4[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 128)          24704       global_max_pooling2d_4[0][0]
__________________________________________________________________________________________________
dense_12 (Dense)                (None, 128)          24704       global_max_pooling2d_4[0][0]
__________________________________________________________________________________________________
age_output (Dense)              (None, 1)            129         dense_10[0][0]
__________________________________________________________________________________________________
race_output (Dense)             (None, 5)            645         dense_11[0][0]
__________________________________________________________________________________________________
gender_output (Dense)           (None, 2)            258         dense_12[0][0]
==================================================================================================
Total params: 724,360
Trainable params: 723,080
Non-trainable params: 1,280
```

Model Performance :

Train Accuracy: 61%
Validation Accuracy: 41%

I did run this model for 1 epoch (Due to time constraints). This Model takes more time to run. If we run this model for more than 15 epochs then we can predict even more accuracy.

Reference:

1. CAB420: Machine Learning Tutorials and Learning Materials

2. Qassim, H., Verma, A., & Feinzimer, D. (2018, January). Compressed residual-VGG16 CNN model for big data places image recognition. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 169-175). IEEE.

3. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.