INST 737 - Intro To Data Science

Milestone 3

Bhavesh Bellara

Gaurav Hasija

Kanishka Jain

Danish Mir

Research Question:

1. Predict the **time duration in days** it takes for a case to be certified or denied.
2. Predict the **case decision.**
3. Predict whether the **agent is present** or not for the case (Random forest).
4. Predict **Threshold Hourly Salary** based on Employee hourly salary and Occupation.

Question 1 - SVM

We use SVM to answer 3 research questions.

1. Predicting the case status (Binary)
2. Predicting if the agent is present or not (Binary)
3. Predicting the duration (Multiclass)

All the research questions were answered based on a sample of 80,000 records in the test set and 20,000 records in the training set. For all the questions, both linear and well as non-linear kernels were used.

1. **Predicting the case status**
   - **Vanilladot kernel**

Command

```
model<-
ksvm(CASE_STATUS_1.0~DURATION+WAGE_RATE_OF_PAY_FROM_HOUR+HOU
RLY_WAGE+OCCUPATION,data=train,kernel="vanilladot")
```

Confusion matrix

pred	0	1
0	430	0
1	389	19181

Confusion matrix tells us that:

- CASE_STATUS=0 was predicted as CASE_STATUS=0 430 times
- CASE_STATUS=1 was predicted as CASE_STATUS=0 0 times
- CASE_STATUS=0 was predicted as CASE_STATUS=1 389 times
- CASE_STATUS=1 was predicted as CASE_STATUS=1 19181 times

Accuracy

Agreement	
FALSE	TRUE
389	19611

According to the table above, 19611 occurrences were predicted correctly and 389 occurrences were predicted incorrectly. So the accuracy of our model is **0.98055** i.e 98.05%

- **Rbfdot kernel**

Command

```
modelBinaryNonLinearCaseStatus<-
ksvm(CASE_STATUS_1.0~DURATION+WAGE_RATE_OF_PAY_FROM_HOU
R+HOURLY_WAGE+OCCUPATION,data=train,kernel="rbfdot")
```

Confusion matrix

pred	0	1
0	462	0
1	357	19181

Confusion matrix tells us that:

- CASE_STATUS=0 was predicted as CASE_STATUS=0 462 times
- CASE_STATUS=1 was predicted as CASE_STATUS=0 0 times
- CASE_STATUS=0 was predicted as CASE_STATUS=1 357 times
- CASE_STATUS=1 was predicted as CASE_STATUS=1 19181 times

Accuracy

Agreement	
FALSE	TRUE
357	19643

According to the table above, 19643 occurrences were predicted correctly and 357 occurrences were predicted incorrectly. So the accuracy of our model is **0.98215** i.e 98.21%

We see that the accuracy with the rbfdot model is slightly more than the accuracy with vanilladot model.

2. Predicting if the Agent is present or not

- **Vanilladot kernel**

Command

```
modelBinaryLinearAgentPresent<-
ksvm(AGENT_PRESENT_1.0~DURATION+WAGE_RATE_OF_PAY_FROM_H
OUR+HOURLY_WAGE+OCCUPATION,data=train,kernel="vanilladot")
```

Confusion matrix

pred	0	1
0	113	71
1	7375	18954

Confusion matrix tells us that:

- AGENT_PRESENT=0 was predicted as AGENT_PRESENT=0 113 times
- AGENT_PRESENT=1 was predicted as AGENT_PRESENT=0 71 times
- AGENT_PRESENT=0 was predicted as AGENT_PRESENT=1 7375 times
- AGENT_PRESENT=1 was predicted as AGENT_PRESENT=1 12441 times

Accuracy

Agreement	
FALSE	TRUE
7446	12554

According to the table above, 12554 occurrences were predicted correctly and 7446 occurrences were predicted incorrectly. So the accuracy of our model is **0.6277** i.e 62.77%

- **Rbfdot kernel**

Command

```
modelBinaryNonLinearCaseStatus<-
ksvm(CASE_STATUS_1.0~DURATION+WAGE_RATE_OF_PAY_FROM_HOU
R+HOURLY_WAGE+OCCUPATION,data=train,kernel="rbfdot")
```

Confusion matrix

pred	0	1
0	4790	3854
1	2780	8576

Confusion matrix tells us that:

- AGENT_PRESENT=0 was predicted as AGENT_PRESENT=0 4790 times
- AGENT_PRESENT=1 was predicted as AGENT_PRESENT=0 3854 times
- AGENT_PRESENT=0 was predicted as AGENT_PRESENT=1 2780 times
- AGENT_PRESENT=1 was predicted as AGENT_PRESENT=1 8576 times

Accuracy

Agreement	
FALSE	TRUE
6634	13366

According to the table above, 13366 occurrences were predicted correctly and 6634 occurrences were predicted incorrectly. So the accuracy of our model is **0.6683** i.e 66.83%

3. **Predicting the Duration (range)**

- **Vanilladot kernel**

Command

modelMulticlassLinearDuration<-
ksvm(DURATION_RANGE~.,data=train,kernel="vanilladot")

Confusion matrix

pred	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
0-10	19298	0	0	0	0	0	0	0	0	0
10-20	0	213	0	0	0	0	0	0	0	0
20-30	0	1	41	0	0	0	0	0	0	0
30-40	0	0	0	40	1	0	0	0	0	0
40-50	0	0	0	0	37	1	0	0	0	0
50-60	0	0	0	0	1	32	0	0	0	0
60-70	0	0	0	0	0	0	32	0	0	0
70-80	0	0	0	0	0	0	0	23	0	0
80-90	0	0	0	0	0	0	0	0	19	0
90-100	0	0	0	0	0	0	0	0	0	0

Confusion matrix tells us that:

- DURATION_RANGE=(0,10] was predicted as DURATION_RANGE=(0,10] 19298 times
- DURATION_RANGE=(10,20] was predicted as DURATION_RANGE=(10,20] 213 times
- DURATION_RANGE=(20,30] was predicted as DURATION_RANGE=(20,30] 41 times
- DURATION_RANGE=(20,30] was predicted as DURATION_RANGE=(10,20] 1 times
- DURATION_RANGE=(30,40] was predicted as DURATION_RANGE=(30,40] 40 times

- DURATION_RANGE=(30,40] was predicted as DURATION_RANGE=(40,50] 1 time
- DURATION_RANGE=(40,50] was predicted as DURATION_RANGE=(40,50] 37 times
- DURATION_RANGE=(40,50] was predicted as DURATION_RANGE=(50,60] 1 time
- DURATION_RANGE=(50,60] was predicted as DURATION_RANGE=(50,60] 32 times
- DURATION_RANGE=(50,60] was predicted as DURATION_RANGE=(40,50] 1 time
- DURATION_RANGE=(60,70] was predicted as DURATION_RANGE=(60,70] 32 times
- DURATION_RANGE=(70,80] was predicted as DURATION_RANGE=(70,80] 23 times
- DURATION_RANGE=(80,90] was predicted as DURATION_RANGE=(80,90] 20 times

Accuracy

Agreement	
FALSE	TRUE
4	19735

According to the table above, 19738 occurrences were predicted correctly and 23 occurrences were predicted incorrectly. So the accuracy of our model is **0.98675** i.e 98.675%

Rbfdot kernel

Command

```
modelMulticlassNonLinearDuration<-
ksvm(DURATION_RANGE~.,data=train,kernel="rbfdot")
```

Confusion matrix

pred	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
0-10	19312	4	0	0	0	0	0	0	0	0
10-20	2	204	1	0	0	0	0	0	0	0
20-30	0	3	70	1	0	0	0	0	0	0
30-40	0	0	1	36	1	0	0	0	0	0
40-50	0	0	0	1	36	1	0	0	0	0
50-60	0	0	0	0	1	21	4	0	0	0
60-70	0	0	0	0	0	2	18	0	0	0
70-80	0	0	0	0	0	0	0	21	1	0
80-90	0	0	0	0	0	0	0	0	20	0
90-100	0	0	0	0	0	0	0	0	0	0

Confusion matrix tells us that:

- DURATION_RANGE=(0,10] was predicted as DURATION_RANGE=(0,10] 19312 times
- DURATION_RANGE=(0,10] was predicted as DURATION_RANGE=(10,20] 4 times
- DURATION_RANGE=(10,20] was predicted as DURATION_RANGE=(10,20] 204 times
- DURATION_RANGE=(10,20] was predicted as DURATION_RANGE=(0,10] 2 times
- DURATION_RANGE=(10,20] was predicted as DURATION_RANGE=(20,30] 1 time
- DURATION_RANGE=(20,30] was predicted as DURATION_RANGE=(20,30] 70 times
- DURATION_RANGE=(20,30] was predicted as DURATION_RANGE=(10,20] 3 times
- DURATION_RANGE=(20,30] was predicted as DURATION_RANGE=(30,40] 1 time
- DURATION_RANGE=(30,40] was predicted as DURATION_RANGE=(30,40] 36 times
- DURATION_RANGE=(30,40] was predicted as

DURATION_RANGE=(20,30] 1 time

- DURATION_RANGE=(30,40] was predicted as DURATION_RANGE=(40,50] 1 time
- DURATION_RANGE=(40,50] was predicted as DURATION_RANGE=(40,50] 36 times
- DURATION_RANGE=(40,50] was predicted as DURATION_RANGE=(30,40] 1 time
- DURATION_RANGE=(40,50] was predicted as DURATION_RANGE=(50,60] 1 time
- DURATION_RANGE=(50,60] was predicted as DURATION_RANGE=(50,60] 21 times
- DURATION_RANGE=(50,60] was predicted as DURATION_RANGE=(40,50] 1 time
- DURATION_RANGE=(50,60] was predicted as DURATION_RANGE=(60,70] 4 times
- DURATION_RANGE=(60,70] was predicted as DURATION_RANGE=(60,70] 18 times
- DURATION_RANGE=(60,70] was predicted as DURATION_RANGE=(50,60] 2 times
- DURATION_RANGE=(70,80] was predicted as DURATION_RANGE=(70,80] 21 times
- DURATION_RANGE=(70,80] was predicted as DURATION_RANGE=(80,90] 1 time
- DURATION_RANGE=(80,90] was predicted as DURATION_RANGE=(80,90] 20 times

Accuracy

Agreement	
FALSE	TRUE
23	19738

According to the table above, 19738 occurrences were predicted correctly and 23 occurrences were predicted incorrectly. So the accuracy of our model is **0.9869** i.e 98.69%

The accuracy with linear model or non linear model is almost the same for this question.

Question 2 - Neural Networks

We are using neural network on for below four research questions:

1. Predict whether the agent is present or not for the case
2. Predict the case decision
3. Predict the time duration in days it takes for a case to be certified or denied
4. Predict Threshold Hourly Salary based on Employee hourly salary and Occupation

We started with normalization of the features and converted the variables duration, hourly wage and wage rate which were in the range of 0 to 1.

We applied min-max normalization here and converted all our continuous variables to range from 0 to 1. All the factors like AGENT_PRESENT CASE_STATUS and OCCUPATION which ranged from binary values to categories were also converted to 0 and 1.

For all four research questions, we have applied the model 4 to 6 times by modifying the number of hidden layers, modifying the number of neurons on each hidden layer and changing the activation function from sigmoid(logistic) to tanh and sometimes increased reps to capture better results.

Number of classes in the research question of our model are 2(present/not present) and two out of four research questions are predicting continuous variables.

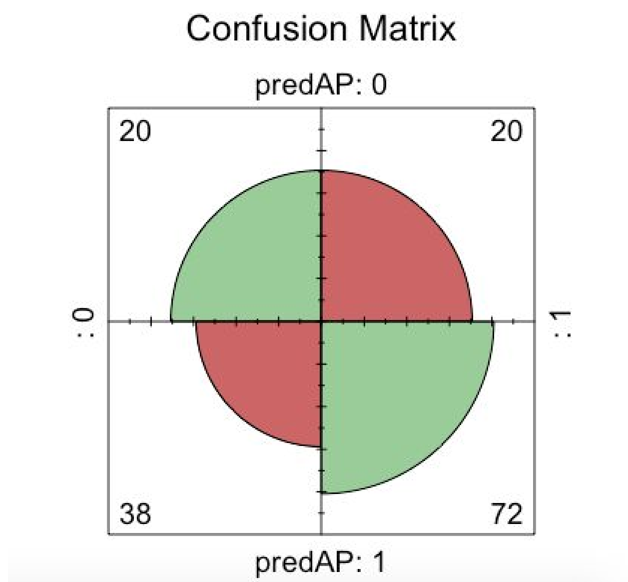
We start now by taking features to neural network model as per the research Question.

1. Predict whether the agent is present or not for the case

Dependent Variable: AGENT_PRESENT_1.0
Independent Variables: WAGE_RATE_OF_PAY_FROM_HOUR+
HOURLY_WAGE
DURATION+ CASE_STATUS_1.0+ DURATION +OCCUPATION_NUM

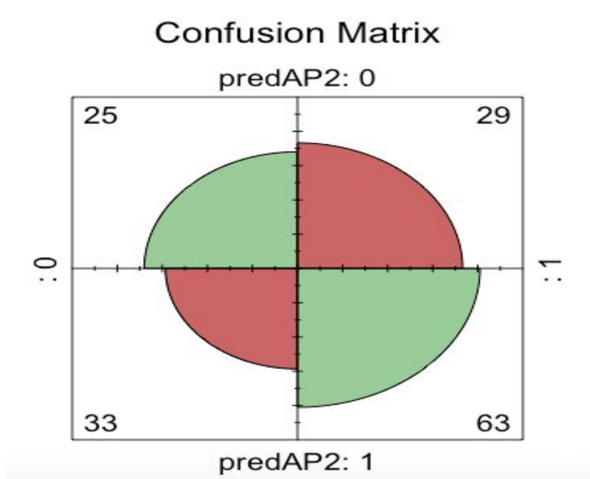
Trial 1:

One hidden layer
Miss classification error: 0.3866667
Confusion Matrix: Here 0 is Agent Not Present and 1 is Agent Present



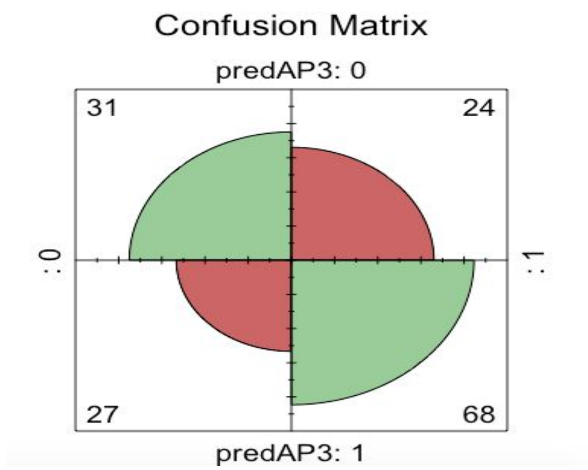
Trial 2:

Two hidden layers
Miss classification error: 0.4133333
Activation function='logistic'
Confusion Matrix: Here 0 is Agent Not Present and 1 is Agent Present



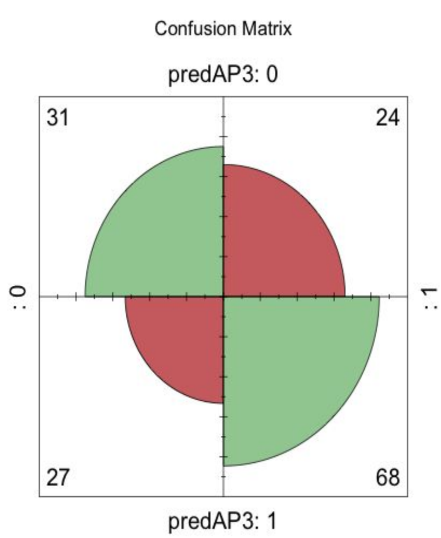
Trial 3:

Two hidden layers: first with two neuron second with 1 neuron
Miss classification error: 0.34
Confusion Matrix: Here 0 is Agent Not Present and 1 is Agent Present



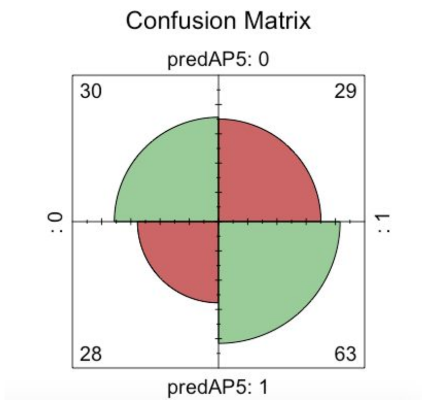
Trial 4: (so far, the best)

Two hidden layers: first with two neuron second with 1 neuron
Miss classification error: 0.33
Activation Function: ‘tanh’
Confusion Matrix: Here 0 is Agent Not Present and 1 is Agent Present



Trial 5:

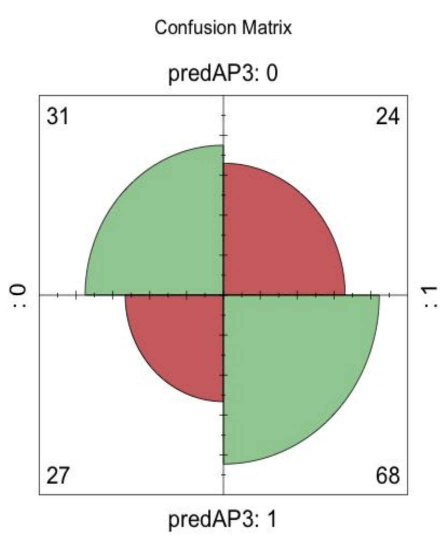
Five hidden layers
Miss classification error: 0.38
Activation Function: ‘tanh’
Confusion Matrix: Here 0 is Agent Not Present and 1 is Agent Present



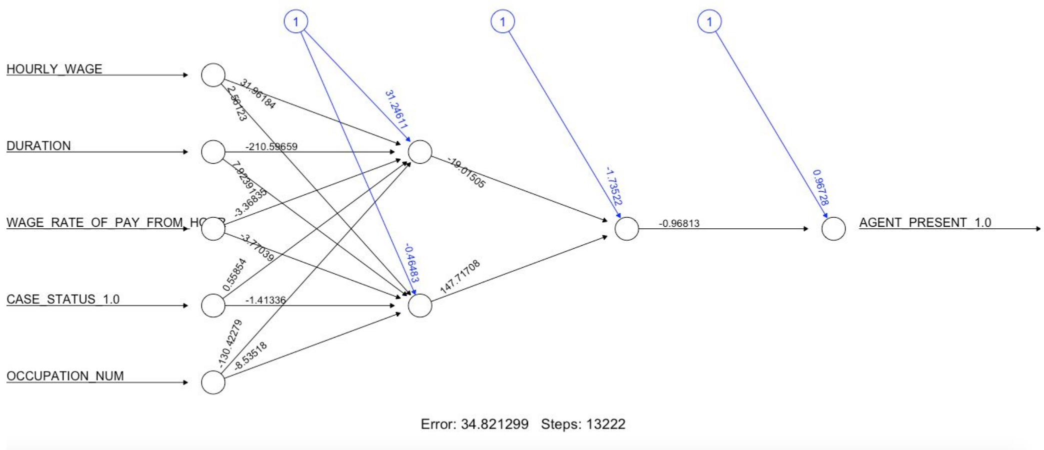
Conclusion

From the above test we found trial number 4 to be the best as miss-classification error was the lowest.

Two hidden layers: first with two neuron second with 1 neuron
Miss classification error: 0.33
Activation Function: ‘tanh’
Confusion Matrix: Here 0 is Agent Not Present and 1 is Agent Present



Best Model:

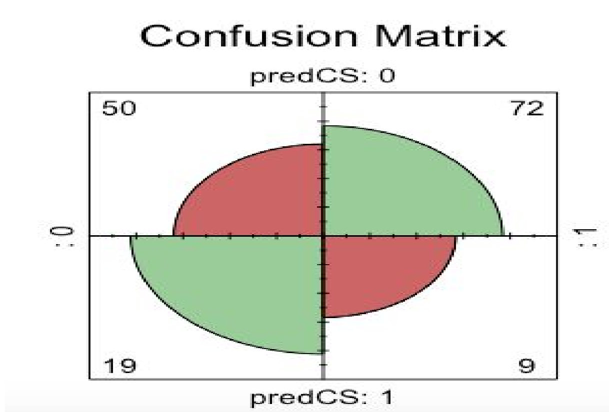


2. Predict the case decision

Dependent Variable: CASE_STATUS_1.0
Independent Variables: WAGE_RATE_OF_PAY_FROM_HOUR+ HOURLY_WAGE DURATION+AGENT_PRESENT_1.0+ DURATION +OCCUPATION_NUM

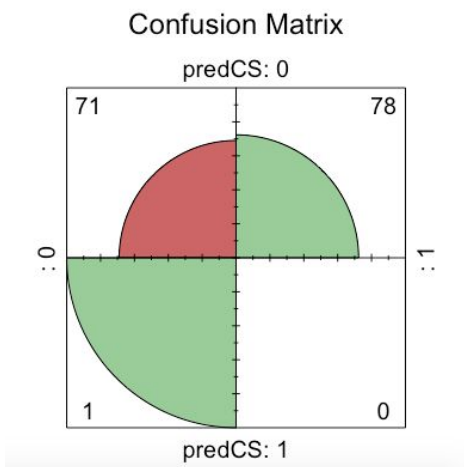
Trial 1:

one hidden layer
Miss classification error: 0.6066667
Activation Function: 'logistic'
Confusion Matrix: Here 0 is Case denied and 1 is case certified.



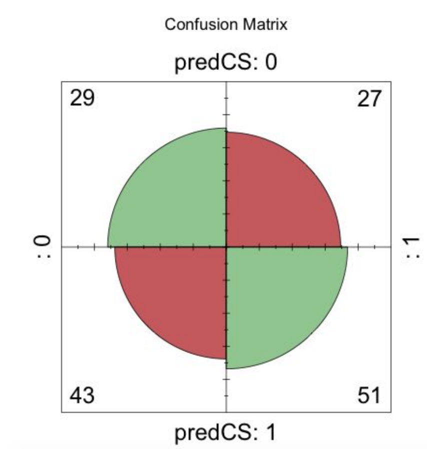
Trial 2:

Two hidden layers
Miss classification error: 0.5266667
Activation Function: ‘tanh
Confusion Matrix: Here 0 is Case denied and 1 is case certified.



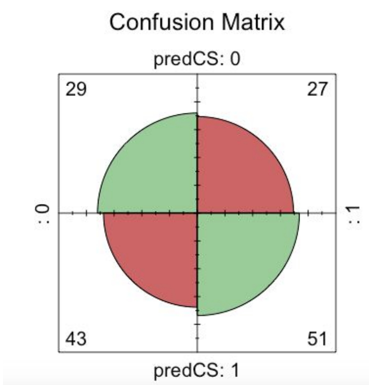
Trial 3:

Two hidden layers: first layer with 2 neuron and second with 1 neuron
Miss classification error: 0.4666667
Activation Function: ‘tanh’
Confusion Matrix: Here 0 is Case denied and 1 is case certified.



Trial 4: (So far, the best)

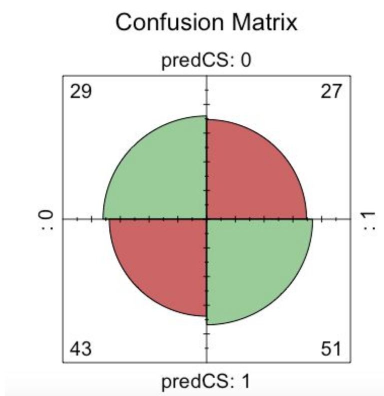
Four hidden layers:
Miss classification error: 0.4
Activation Function: ‘tanh’
Confusion Matrix: Here 0 is Case denied and 1 is case certified.



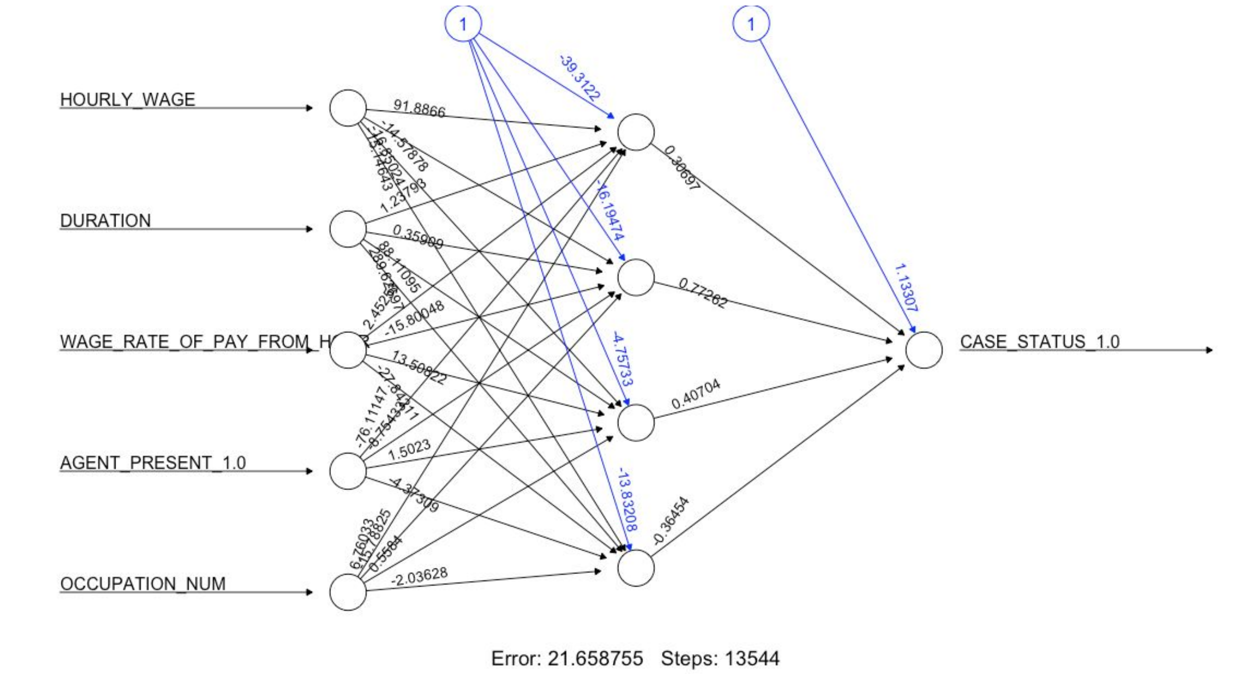
Conclusion

Out of four trials, trial 4 was the best trial with the lowest miss-classification error of 40% hence we will plot this model.

Four hidden layers:
Miss classification error: 0.4
Activation Function: ‘tanh’
Confusion Matrix: Here 0 is Case denied and 1 is case certified.



Best Model:



3. Predict the time duration in days it takes for a case to be certified or denied

Dependent Variable: DURATION
Independent Variables: WAGE_RATE_OF_PAY_FROM_HOUR+ HOURLY_WAGE DURATION+AGENT_PRESENT_1.0+CASE_STATUS_1.0+OCCUPATION_NUM

Trial 1:

> One hidden layer
> Correlation coefficient: 0.1173611
> Activation Function: ‘logistic’

Trial 2:

> Five hidden layers
> Correlation coefficient: 0.02559979
> Activation Function: ‘logistic’

Trial 3:

One hidden layer
Correlation coefficient: 0.03506084
Activation Function: ‘tanh

Trial 4(so far, the best)

Two hidden layers, first with 2 neurons and second with 4 Neurons.
Correlation coefficient: 0.1855026
Activation Function: ‘logistic’

Trial 5:

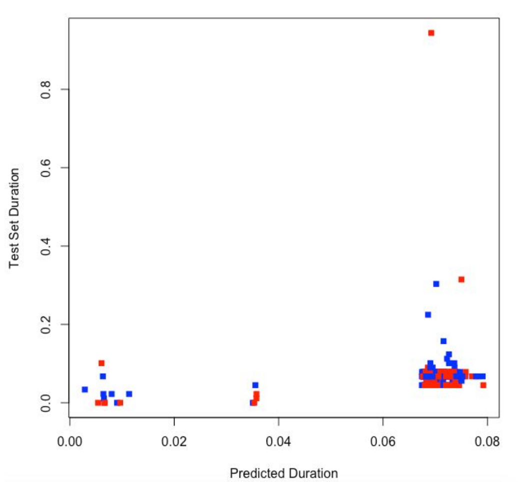
Four hidden layers
Correlation coefficient: 0.0704616
Activation Function: ‘Tanh”

Conclusion

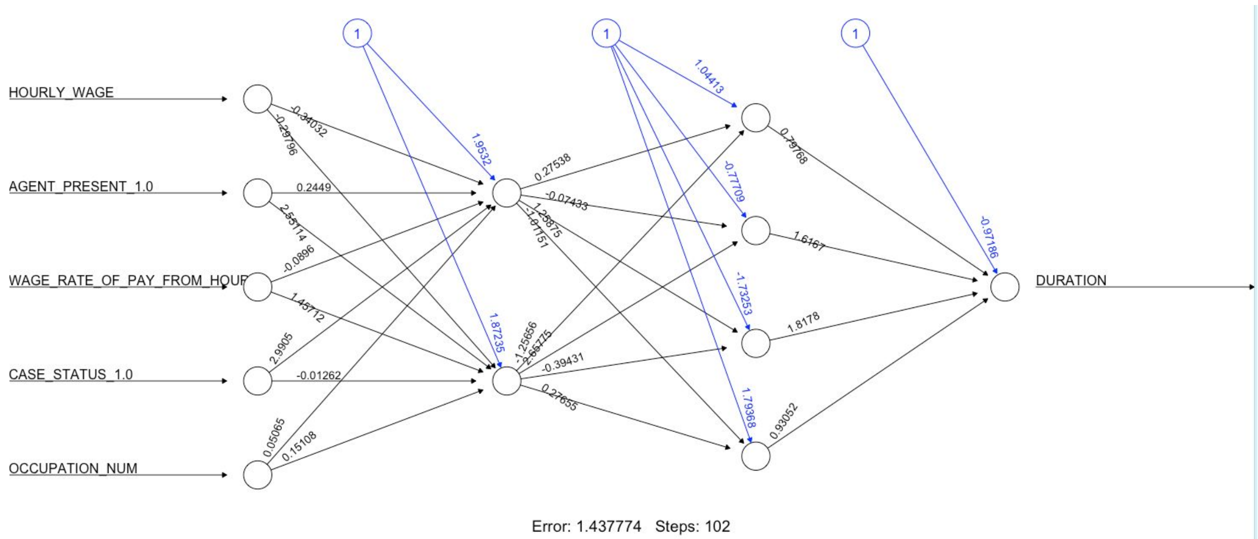
According to above trials, trail 5 is the best results among all as correlation is highest and we will take that as the best model here and plot the results.

Two hidden layers, first with 2 neurons and second with 4 Neurons.
Correlation coefficient: 0.1855026
Activation Function: ‘logistic’

Plotting the predicted duration with test set duration:



Plot of the model:



4. Predict Threshold Hourly Salary based on Employee hourly salary and Occupation

Dependent Variable: HOURLY_WAGE
Independent Variables: WAGE_RATE_OF_PAY_FROM_HOUR+DURATION
DURATION+AGENT_PRESENT_1.0+CASE_STATUS_1.0+OCCUPATION_NUM

Trial 1:

One hidden layer
Correlation coefficient: 0.1959044
Activation Function: ‘Tanh”

Trial 2:

Two hidden layers
Correlation coefficient: 0.1999577
Activation Function: ‘tanh”

Trial 3:

Three hidden layers
Correlation coefficient: 0.2003638
Activation Function: ‘tanh”

Trial 4(So far, the best)

Two hidden layers with 2 neurons in each layer
Correlation coefficient: 0.2082862
Activation Function: ‘tanh”

Trial 5:

Two hidden layers with 2 in first and 1 second hidden layer
Correlation coefficient: 0.2003648
Activation Function: ‘tanh”

Trial 6:

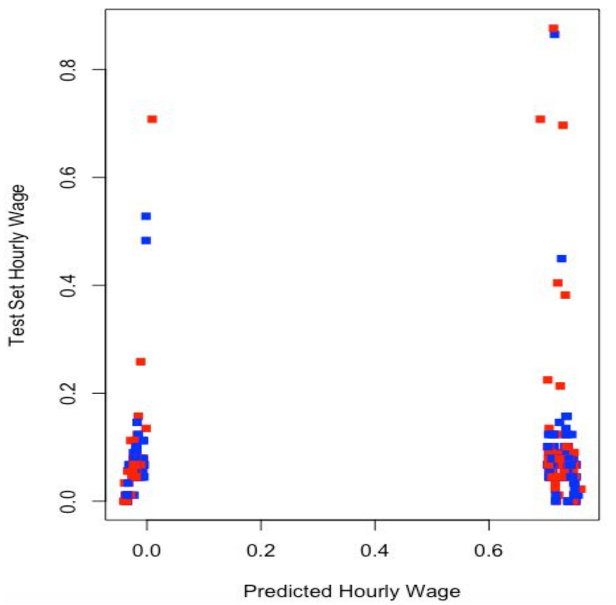
Five hidden layers
Correlation coefficient: 0.06858963
Activation Function: ‘tanh”

Conclusion

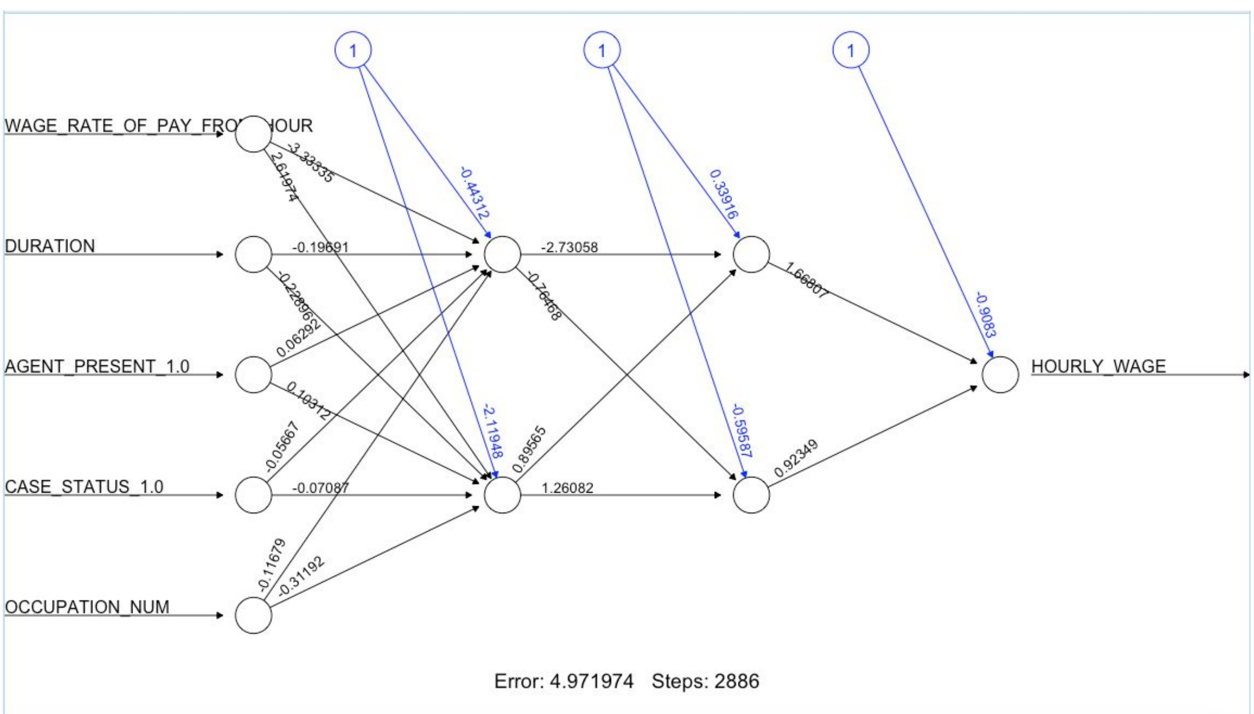
Best results were observed in the trial 4, we had maximum correlation coefficient observed, hence we chose to build the model for the that.

Two hidden layers with 2 neurons in each layer
Correlation coefficient: 0.2082862
Activation Function: ‘tanh”

Plotting predicted hourly wage against test set hourly wage to check the correlation



Best Neural Net model



Question 3 - Clustering

K-Means Clustering:

K-Means is considered one of the simplest models amongst them. Despite its simplicity, the K-means is vastly used for clustering in many data science applications, especially useful if you need to quickly discover insights from unlabeled data.

We used K-means to look into the distribution of Case Status based on their Duration, Hourly wage, and Agent Present or not.

We started with sklearn.cluster and imported kMeans

```
]: import random
   import matplotlib.pyplot as plt
   from sklearn.cluster import KMeans
   %matplotlib inline
```

Preprocessing was done to normalize data over the standard deviation. K-means being un-supervised clustering cannot handle categorical values, we had to get rid of the index and create an array with just values of our data set

```
68]: #Normalizing over the standard deviation
     from sklearn.preprocessing import StandardScaler
     X = cdf.values
     X = np.nan_to_num(X)
     Clus_dataSet = StandardScaler().fit_transform(X)
     Clus_dataSet

68]: array([[-0.05713887,  0.86855142,  1.28222103, -0.20814923,  1.23994631],
            [-0.05713887, -1.42337552,  1.28222103, -0.20814923, -0.93321005],
            [-0.43691626, -0.14335595, -0.77989674, -0.20814923,  1.23994631],
            ...,
            [-1.00658233, -0.09146326,  1.28222103,  4.80424545,  1.23994631],
            [-0.05713887,  2.0534344 , -0.77989674, -0.20814923,  0.15336813],
            [-0.05713887,  0.4707075 ,  1.28222103, -0.20814923,  1.23994631]])
```

Modeling:

```
clusterNum = 3
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
k_means.fit(X)
labels = k_means.labels_
print(labels)

[2 0 2 ... 2 1 2]
```

We used ClusterNum = 3 and n_init = 12

Further, the labels were assigned to each row in the data frame and checked the centroid values by averaging the features in each cluster.

```
cdf.groupby('Clus_km').mean() #checking the centroid values by averaging the features in each cluster.
```

Clus_km	DURATION	HOURLY_WAGE	AGENT_PRESENT_0.0	CASE_STATUS_0.0
0	6.283882	28.844612	0.433076	0.044763
1	6.356020	59.672584	0.158144	0.037336
2	6.304342	40.997385	0.382693	0.038569

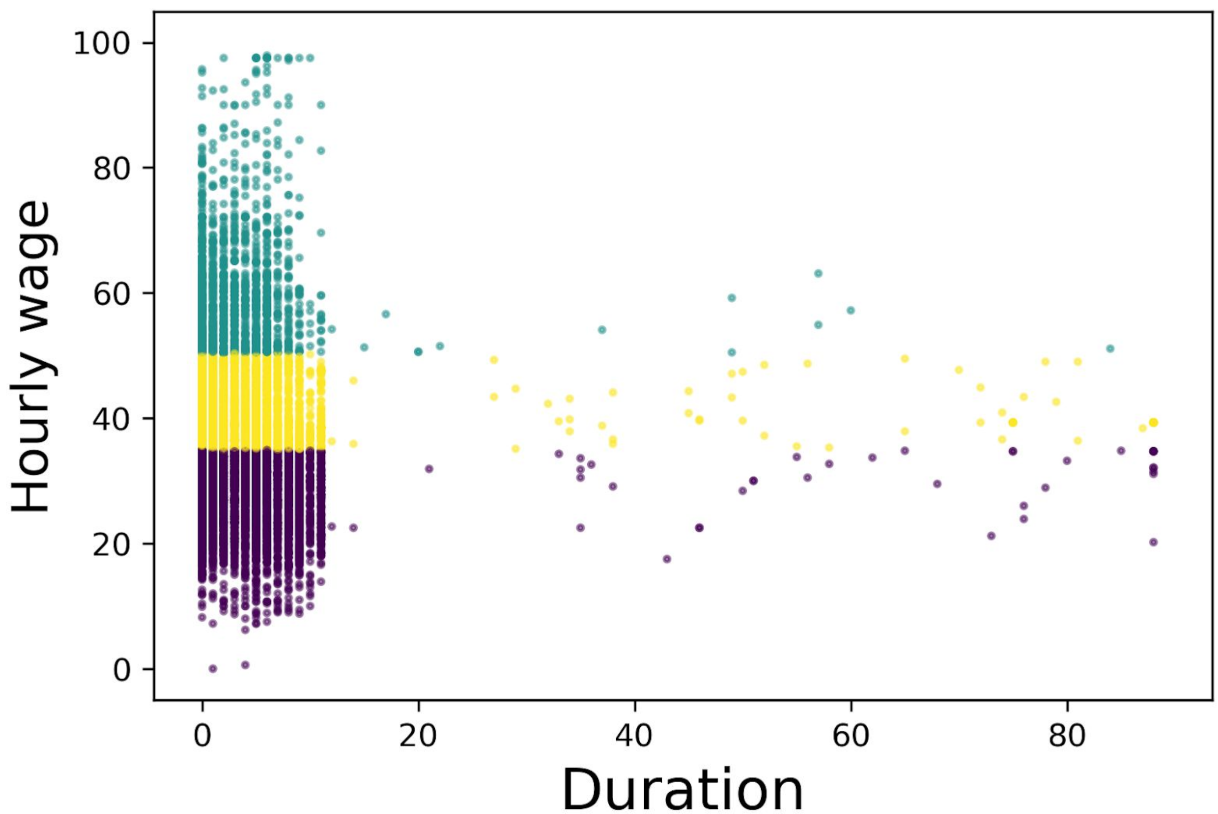


Fig. ()

The figure() above showcases the k-means with three clusters in two dimensions.

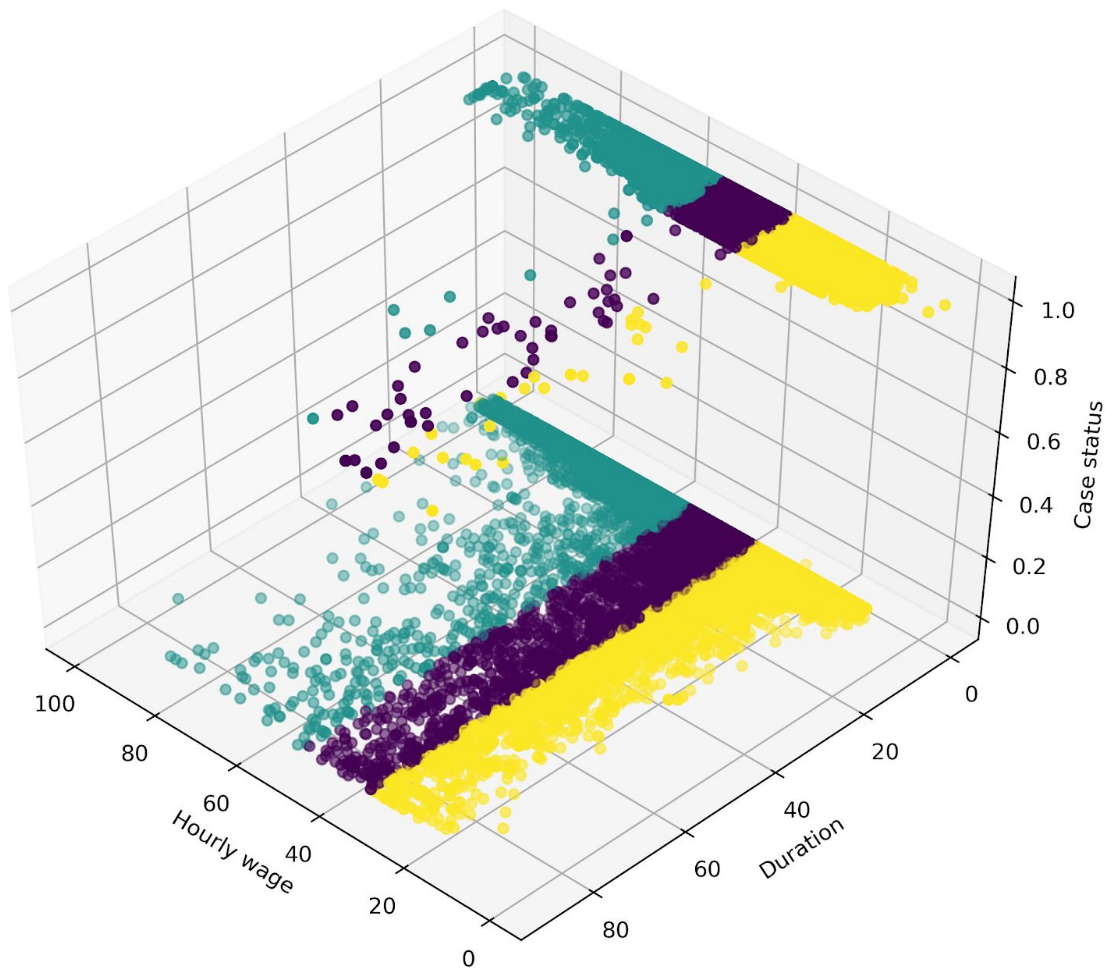


Fig above showcases the k-means clustering in 3d modeling hourly wage, Duration and Case Status.

As can be seen in both the figures K-means clustering is unable to handle different density and irregular shapes in our data set.

Hierarchical clustering:

In hierarchical clustering, we will be using Agglomerative Hierarchical Clustering. The agglomerative is the bottom-up approach and is more popular than Divisive clustering. For our analysis we would use complete Linkage, however, average linkage can also be used.

From our data set, we used DURATION, HOURLY_WAGE, AGENT_PRESENT and CASE_STATUS to perform Hierarchical clustering.

We started with Normalization where we used MinMaxScaler from sklearn.preprocessing to transform features by scaling each feature to a given range. It is by default (0, 1). That is this estimator scales and translates each feature individually such that it is between zero and one.

```
from sklearn.preprocessing import MinMaxScaler
x = cdf.values #returns a numpy array
min_max_scaler = MinMaxScaler()
feature_mtx = min_max_scaler.fit_transform(x)
feature_mtx [0:5]
```

```
array([[0.06818182, 0.43517475, 1.        , 0.        ],
       [0.06818182, 0.13641488, 1.        , 0.        ],
       [0.04545455, 0.30326945, 0.        , 0.        ],
       [0.07954545, 0.24577227, 1.        , 0.        ],
       [0.06818182, 0.28297632, 0.        , 0.        ]])
```

We used scikit-learn to calculate the distance matrix.

```
dist_matrix = distance_matrix(feature_mtx,feature_mtx)
print(dist_matrix)
```

```
[[0.         0.29875986 1.008918   ... 1.015741   0.04058625 1.00766034]
 [0.29875986 0.         1.01407937 ... 1.00724968 0.25817362 1.01515333]
 [1.008918   1.01407937 0.         ... 0.0515084  1.00441809 0.02405845]
 ...
 [1.015741   1.00724968 0.0515084  ... 0.         1.00941462 0.05411499]
 [0.04058625 0.25817362 1.00441809 ... 1.00941462 0.         1.00347402]
 [1.00766034 1.01515333 0.02405845 ... 0.05411499 1.00347402 0.        ]]
```

From the distance matrix we were able to generate the dendrogram. We used scipy.cluster.hierarchy library for this.

```
Z = hierarchy.linkage(dist_matrix, 'complete')
```

```
import pylab
import scipy.cluster.hierarchy
fig = pylab.figure(figsize=(18,50))

dendro = hierarchy.dendrogram(Z, leaf_rotation=0, orientation = 'right')
plt.savefig('dendogram.png', format='png', dpi = 300, orientation = 'landscape', transparent=False, bbox_inches='tigl
```

Further, the labels were assigned to each row in the data frame and checked the centroid values by averaging the features in each cluster.

Next, we used the 'AgglomerativeClustering' function from scikit-learn library to cluster the dataset. The AgglomerativeClustering performs a hierarchical clustering using a bottom-up approach. The linkage criteria determine the metric used for the merge strategy:

```
agglom = AgglomerativeClustering(n_clusters = 6, linkage = 'complete')
agglom.fit(feature_mtx)
agglom.labels_
```

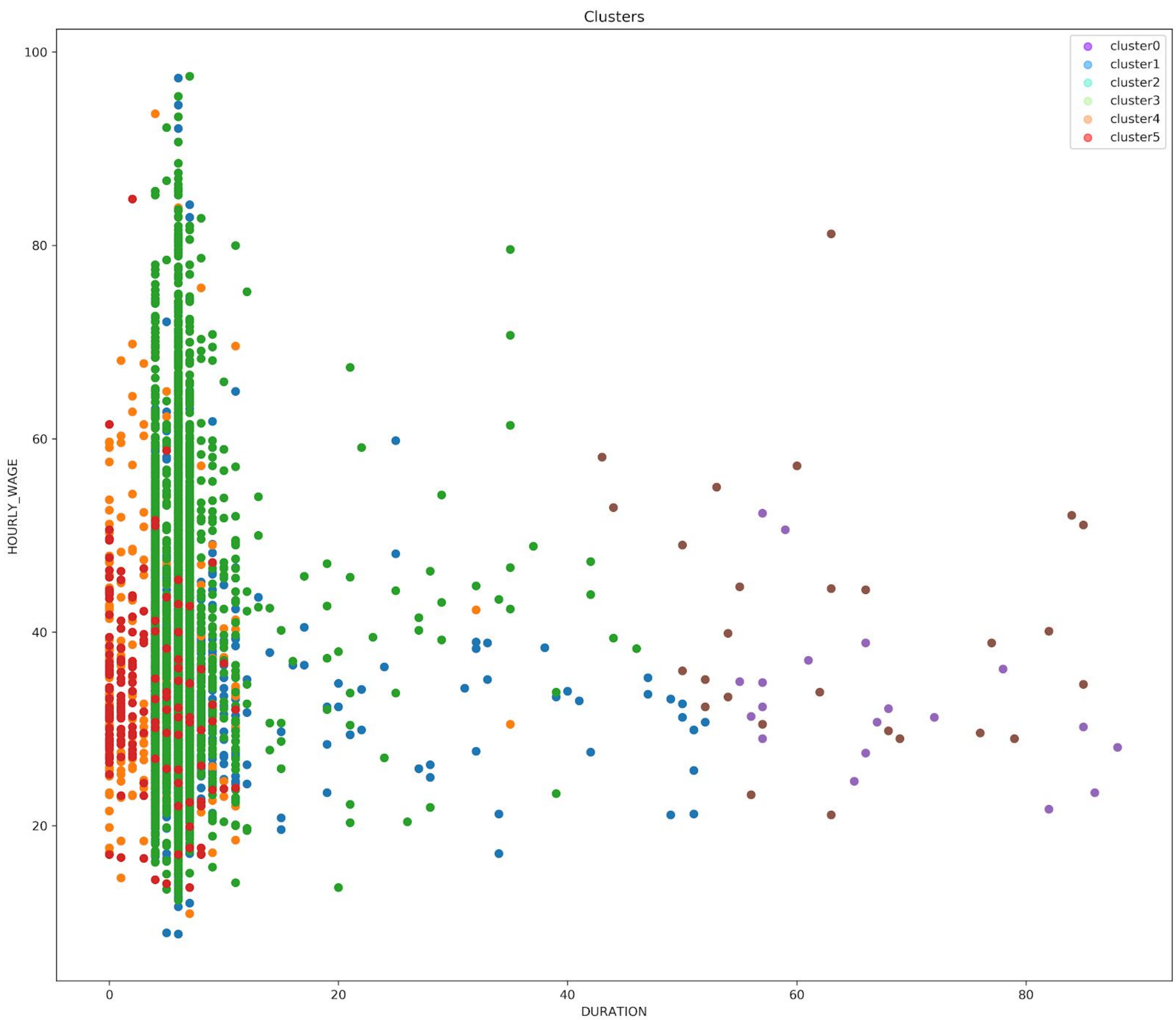
```
array([0, 0, 2, ..., 2, 0, 2])
```

We added a new field to our data frame to show the cluster of each row.

```
cdf['cluster_'] = agglom.labels_
cdf.head()
```

	DURATION	HOURLY_WAGE	AGENT_PRESENT	CASE_STATUS	cluster_
759346	6	47.4	1	0	0
356932	6	20.9	1	0	0
620091	4	35.7	0	0	2
663836	7	30.6	1	0	0
257533	6	33.9	0	0	2

The fig() below showcases the 5 clusters in our model.



As you can see, we are seeing the distribution of each cluster using the scatter plot, but it is not very clear where the centroid of each cluster is. Moreover, there are 2 types of CASE_STATUS in our dataset, "ACCEPTED" (value of 1 in the CASE_STATUS column) and "REJECTED" (value of 0 in the CASE_STATUS column), and likewise is the case with Agent present or not. So, we use them to distinguish the classes and summarize the cluster.

```
cdf.groupby(['cluster_','CASE_STATUS','AGENT_PRESENT'])['cluster_'].count()
```

```
cluster_  CASE_STATUS  AGENT_PRESENT
0         0            1                3443
1         1            0                 263
2         0            0                5690
3         1            1                 177
4         0            1                  19
5         0            0                  27
Name: cluster_, dtype: int64
```

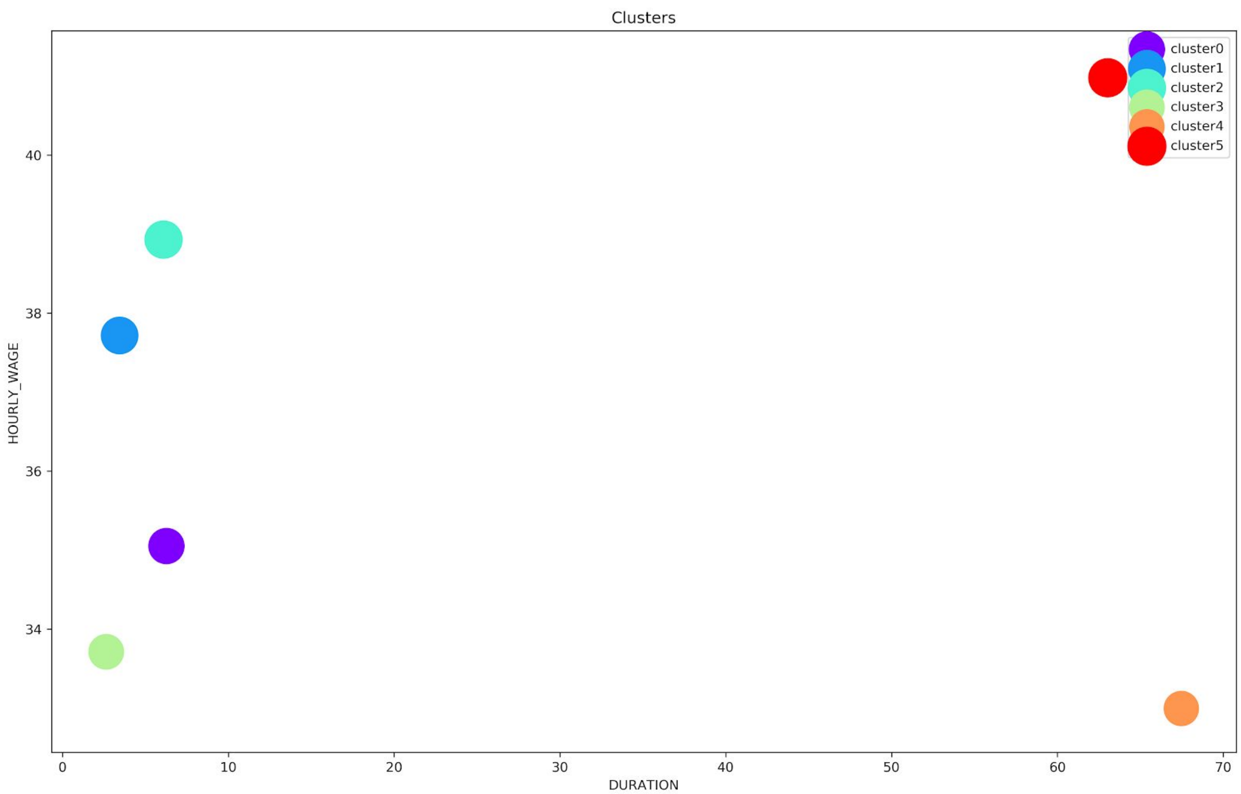
We further grouped our data and looked into means

```
agg = cdf.groupby(['cluster_','CASE_STATUS', 'AGENT_PRESENT'])['DURATION', 'HOURLY_WAGE'].mean()
agg
```

cluster_	CASE_STATUS	AGENT_PRESENT	DURATION	HOURLY_WAGE
0	0	1	6.266047	35.053180
1	1	0	3.444867	37.717110
2	0	0	6.081371	38.931810
3	1	1	2.632768	33.712429
4	0	1	67.473684	32.994737
5	0	0	63.037037	40.977778

From the data above we can see 2 main clusters for Case status 0, viz cluster 4 and cluster 5 with means 67.47 and 63.03.

For case status 1 the cluster 1 and cluster 3 seem to have the majority of data points. Hourly wage seems to be comparatively uniformly distributed among clusters. For Agent Present 0, cluster 5 seems to have majority data points for 'Duration" and Agent present 1, cluster 4. Hourly wage again seems to be uniformly distributed among clusters.



The Fig() above showcases the clusters in our agglomerative clustering.

Density-Based Clustering

Above we saw, k-means, hierarchical can be used to group data without supervision. However, when applied to tasks with arbitrary shape clusters, or clusters within clusters, the traditional techniques might be unable to achieve good results. That is, elements in the same cluster might not share enough similarity or the performance may be poor. Density-based Clustering, under such, locates regions of high density that are separated from one another by regions of low density. Density, in this context, is defined as the number of points within a specified radius.

We again started with our data set consisting of DURATION, HOURLY_WAGE, AGENT_PRESENT and CASE_STATUS to perform DBSCN clustering.

Modelling:

DBSCAN is Density-Based Spatial Clustering of Applications with Noise. This technique is one of the most common clustering algorithms which works based on density of object. The whole idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster.

It works based on two parameters: Epsilon and Minimum Points

Epsilon determine a specified radius that if includes enough number of points within, we call it dense area

minimumSamples determine the minimum number of data points we want in a neighborhood to define a cluster.

We used the sklearn library to run DBSCAN clustering from a vector array or distance matrix. In our case, we pass it the Numpy array dataSet to find core samples of high density and expand clusters from them.

```
from sklearn.cluster import DBSCAN
import sklearn.utils
from sklearn.preprocessing import StandardScaler
sklearn.utils.check_random_state(1000)
Clus_dataSet = cdf[['DURATION','HOURLY_WAGE']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)
```

```
# Compute DBSCAN
db = DBSCAN(eps=0.15, min_samples=10).fit(Clus_dataSet)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
cdf["cluster_"]=labels
```

```
realClusterNum=len(set(labels)) - (1 if -1 in labels else 0)
clusterNum = realClusterNum
print(clusterNum)
```

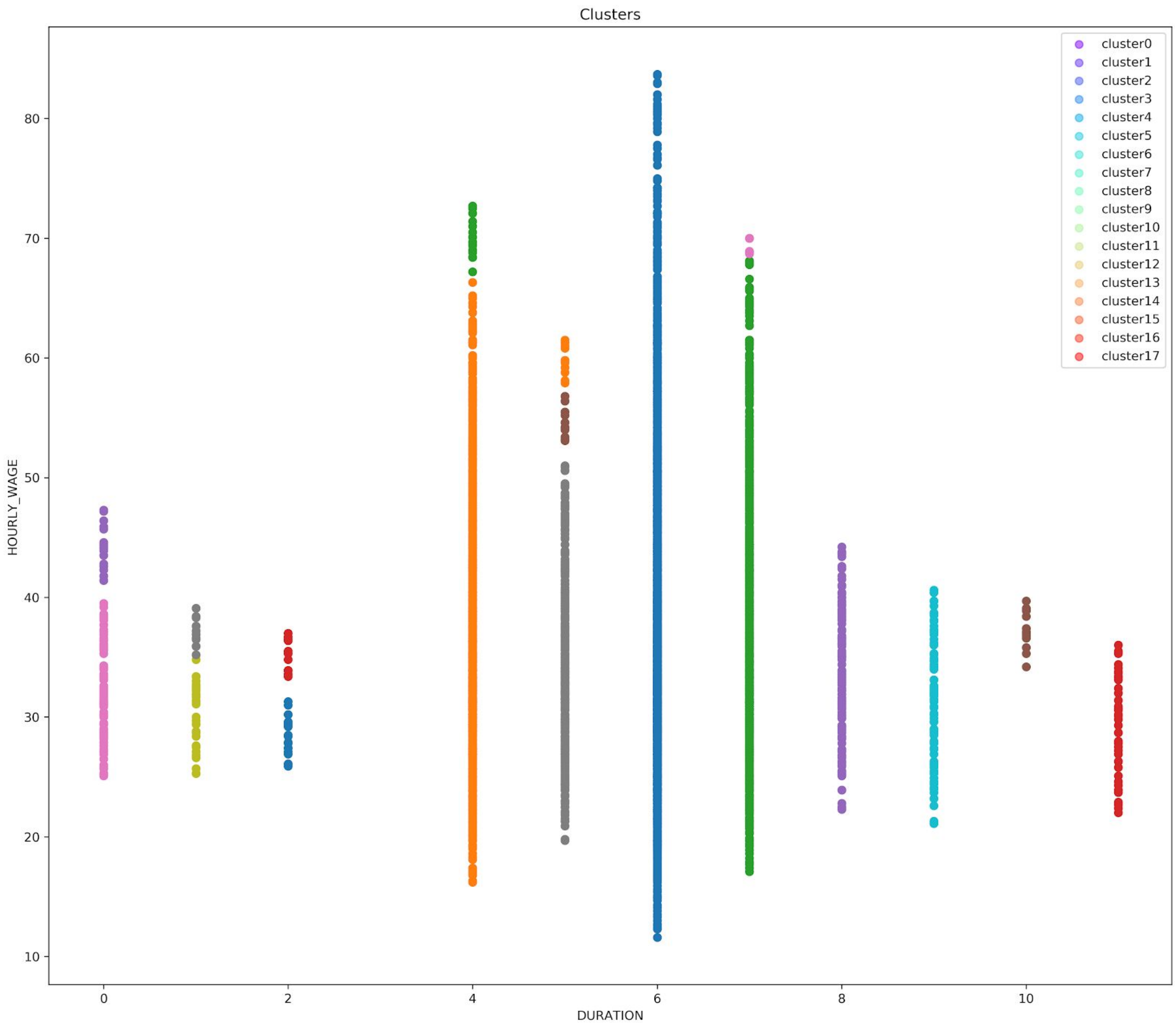
18

```
# A sample of clusters
cdf.head(5)
```

	DURATION	HOURLY_WAGE	AGENT_PRESENT	CASE_STATUS	cluster_
759346	6	47.4	1	0	0
356932	6	20.9	1	0	0
620091	4	35.7	0	0	1
663836	7	30.6	1	0	2
257533	6	33.9	0	0	0

As you can see for outliers, the cluster label is -1

The Fig() below showcases our clusters



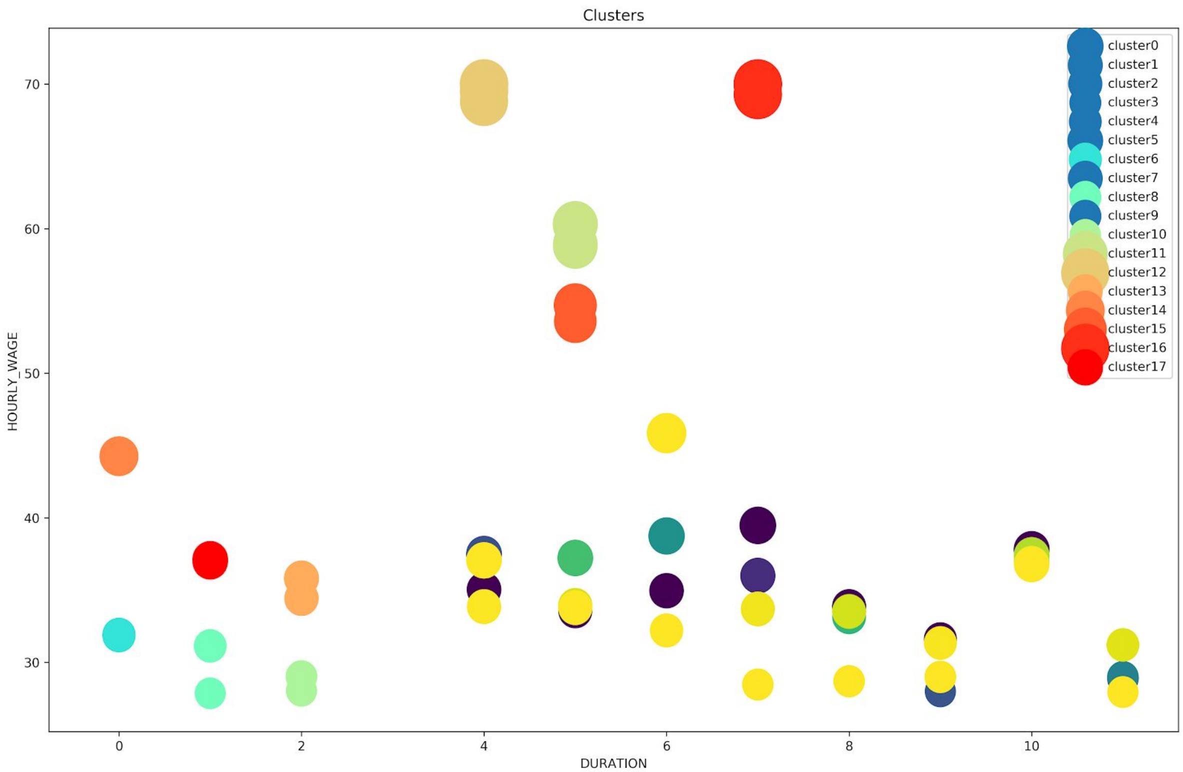
Again, as we are seeing the distribution of each cluster using the scatter plot, it is not very clear where the centroid of each cluster is. Moreover, there are 2 types of CASE_STATUS in our dataset, "ACCEPTED" (value of 1 in the CASE_STATUS column) and "REJECTED" (value of 0 in the CASE_STATUS column), and likewise is the case with Agent present or not. So, we use them to distinguish the classes and summarize the cluster.

We grouped by our data as per 'cluster_','CASE_STATUS', 'AGENT_PRESENT', mean of 'DURATION', 'HOURLY_WAGE'.

```
agg = cdf.groupby(['cluster_','CASE_STATUS', 'AGENT_PRESENT'])['DURATION', 'HOURLY_WAGE'].mean()
agg
```

			DURATION	HOURLY_WAGE
cluster_	**CASE_STATUS**	**AGENT_PRESENT**		
-1	0	0	19.073733	49.842396
		1	28.738739	36.595495
	1	0	3.988235	41.584706
		1	2.842105	36.068421
0	0	0	6.000000	38.746542
		1	6.000000	34.959776
	1	0	6.000000	45.857143
		1	6.000000	32.215385
1	0	0	4.000000	37.492280
		1	4.000000	35.041021
	1	0	4.000000	33.858824
		1	4.000000	37.060000
2	0	0	7.000000	39.477647
		1	7.000000	35.998009
	1	0	7.000000	33.700000
		1	7.000000	28.471429
3	0	0	11.000000	28.904348
		1	11.000000	28.981250

The fig() showcases our 18 clusters in our model.



Question 4 - Comparative Analysis

We applied three cross validation methods on the classifiers using the caret package in R.

1. K-Fold
2. Boot
3. LGOCV

Following accuracy results were obtained:

AGENT_PRESENT			
	K-Fold	LGOCV	Boot
SVM	61.91%	62.2%	62.7%
NaiveBayes	62.82%	61.75%	62.05%
Decision Tree	64.25%	65.25%	60.2%
Logistic Regression	60.88%	58%	61.65%

CASE_STATUS			
	K-Fold	LGOCV	boot
SVM	97.7%	97.1%	98%
NaiveBayes	97.8%	97.3	97.8%
Decision Tree	97.6%	98.1%	96.5%
Logistic Regression	97.6%	96.1%	97%

Feature Selection for improved performance

https://go.umd.edu/5kK

Feature Selection was done based on the above url and results of the test performed found there are no two independent variables with more than 75% correlation, hence all the below features were taken as input to models

	AGENT_PRESENT_1.0	DURATION	WAGE_RATE_OF_PAY_FROM_HOUR	CASE_STATUS_1.0	OCCUPATION_NUM
AGENT_PRESENT_1.0	1.00000000	-0.01256590	0.16344344	0.02715279	0.09456560
DURATION	-0.01256590	1.00000000	0.02808794	0.14735846	-0.08110701
WAGE_RATE_OF_PAY_FROM_HOUR	0.16344344	0.02808794	1.00000000	-0.06122792	-0.11758383
CASE_STATUS_1.0	0.02715279	0.14735846	-0.06122792	1.00000000	-0.09546458
OCCUPATION_NUM	0.09456560	-0.08110701	-0.11758383	-0.09546458	1.00000000