Tecnológico de Monterrey

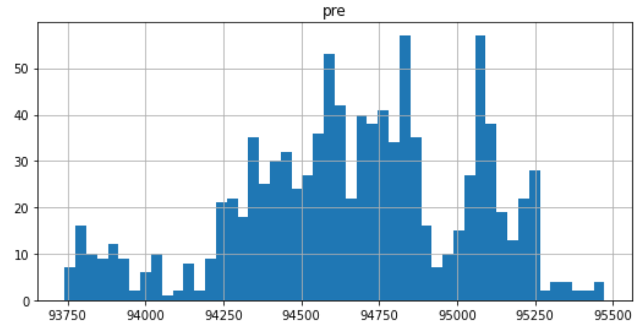
Machine Learning
1st Partial Exam

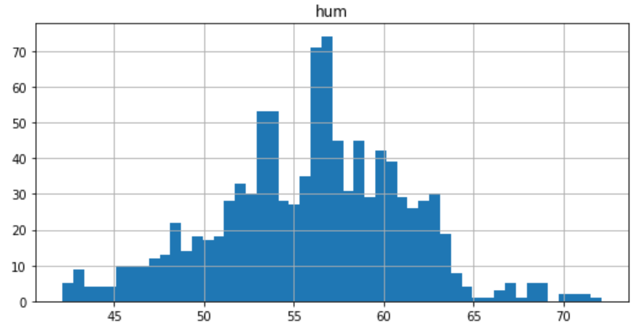
Roberto Ramírez Monroy	A01366943
Alitzel Macías Infante	A01373166

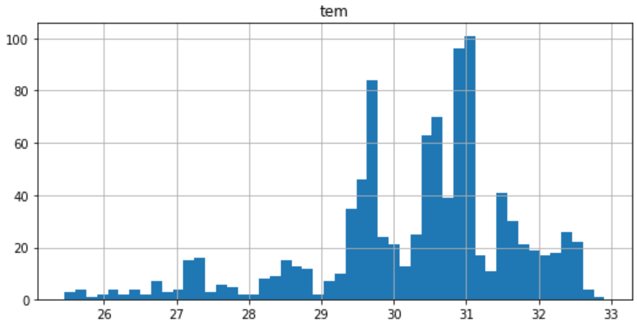
September, 2020

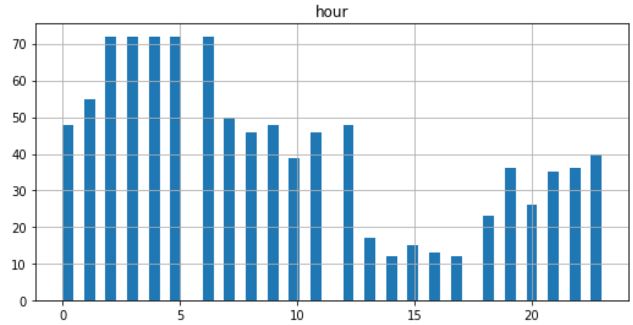
1. Data Analysis

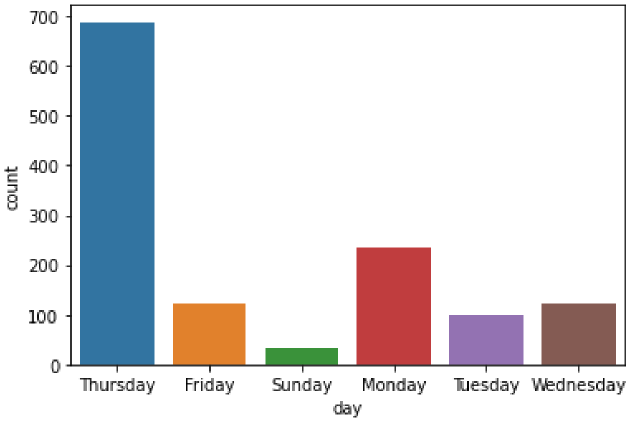
a) *Statistical Distribution of the features.*

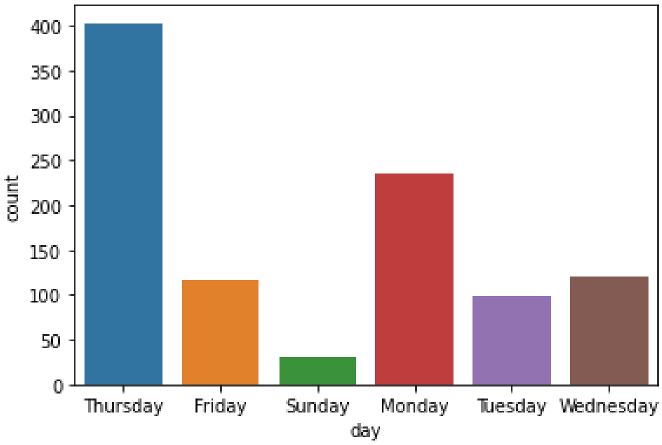




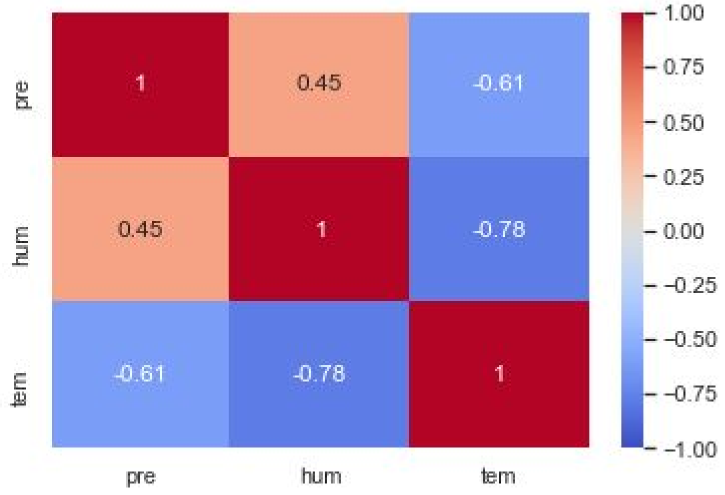








b) *Correlation matrix:*



2. Models

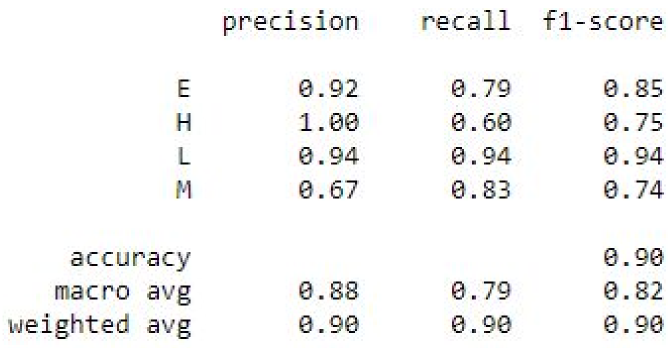
2.1. Decision Trees

For these models we did a dataset split of 80% for training and 20% for testing. The features used in these models were 'tem', 'pre', and 'hum' (temperature, pressure and humidity). The parameters changed between models were the *criterion* ('gini' or 'entropy') and the *max_depth*. Then, we evaluated each model through its *accuracy*, *F-Score*, *precision*, *recall*, and its *confusion matrix*.

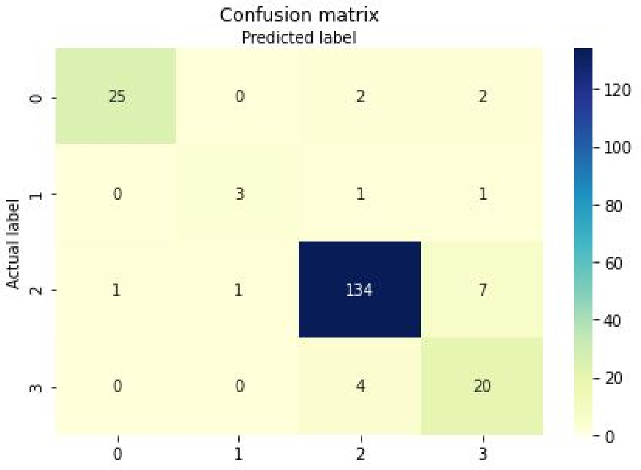
The results of some of the runned models are:

a) *Criterion*: Gini, max_depth: none selected.
Using the parameters specified above, the average accuracy after cross validation through running the model and the dataset partition 50 times was:

Accuracy: 0.89602
The evaluation of one of the runned models is:



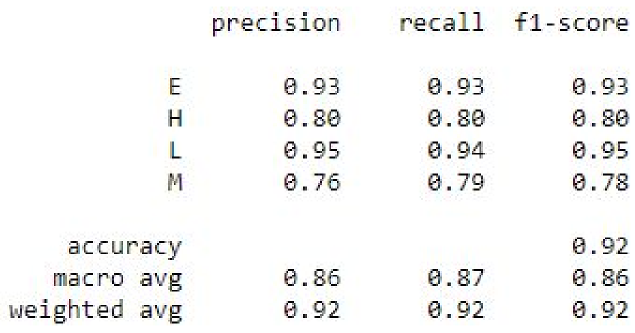
```
              precision    recall  f1-score

           E       0.92      0.79      0.85
           H       1.00      0.60      0.75
           L       0.94      0.94      0.94
           M       0.67      0.83      0.74

    accuracy                           0.90
   macro avg       0.88      0.79      0.82
weighted avg       0.90      0.90      0.90
```



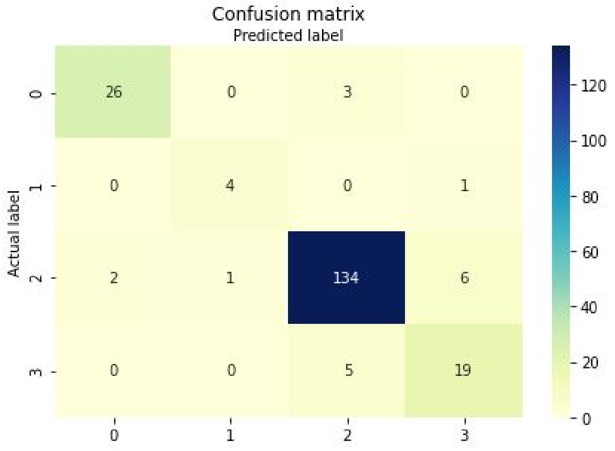
	0	1	2	3
0	25	0	2	2
1	0	3	1	1
2	1	1	134	7
3	0	0	4	20

b) *Criterion*: Entropy, max_depth: none selected.
Using the parameters specified above, the average accuracy after cross validation through running the model and the dataset partition 50 times was:

Accuracy: 0.91857.
The evaluation of one of the runned models is:



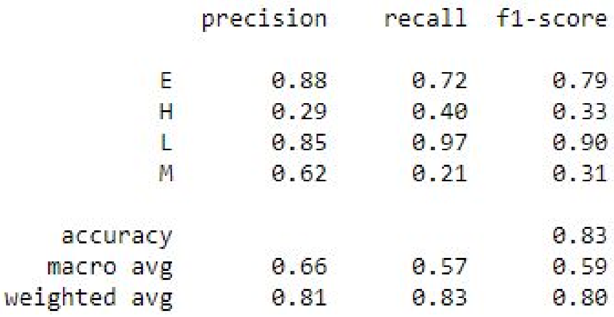
```
              precision    recall  f1-score

           E       0.93      0.93      0.93
           H       0.80      0.80      0.80
           L       0.95      0.94      0.95
           M       0.76      0.79      0.78

    accuracy                           0.92
   macro avg       0.86      0.87      0.86
weighted avg       0.92      0.92      0.92
```



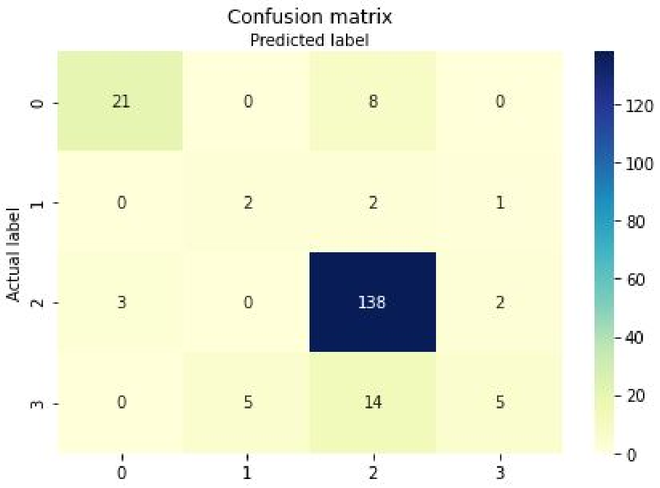
	0	1	2	3
0	26	0	3	0
1	0	4	0	1
2	2	1	134	6
3	0	0	5	19

c) *Criterion*: Gini, max_depth: 5.
Using the parameters specified above, the average accuracy after cross validation through running the model and the dataset partition 50 times was:

Accuracy: 0.82587.
The evaluation of one of the runned models is:



```
              precision    recall  f1-score

           E       0.88      0.72      0.79
           H       0.29      0.40      0.33
           L       0.85      0.97      0.90
           M       0.62      0.21      0.31

    accuracy                           0.83
   macro avg       0.66      0.57      0.59
weighted avg       0.81      0.83      0.80
```

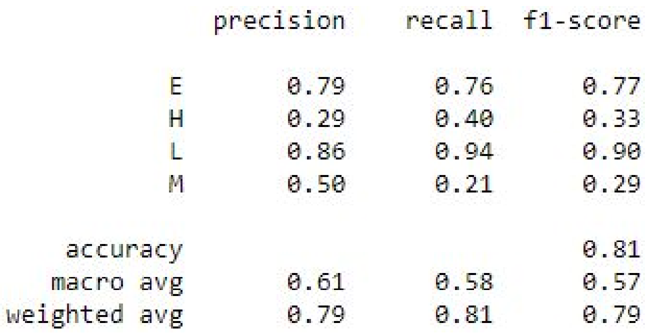


	0	1	2	3
0	21	0	8	0
1	0	2	2	1
2	3	0	138	2
3	0	5	14	5

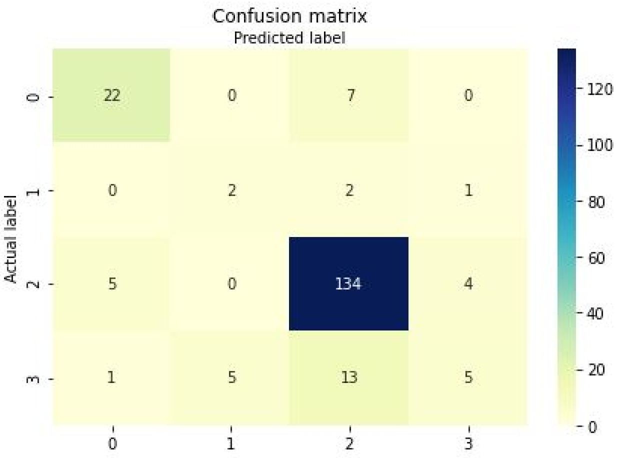
d) *Criterion*: Entropy, max_depth: 5.
Using the parameters specified above, the average accuracy after cross validation through running the model and the dataset partition 50 times was:

Accuracy: 0.81094.

The evaluation of one of the runned models is:



```
              precision    recall  f1-score

           E       0.79      0.76      0.77
           H       0.29      0.40      0.33
           L       0.86      0.94      0.90
           M       0.50      0.21      0.29

    accuracy                           0.81
   macro avg       0.61      0.58      0.57
weighted avg       0.79      0.81      0.79
```

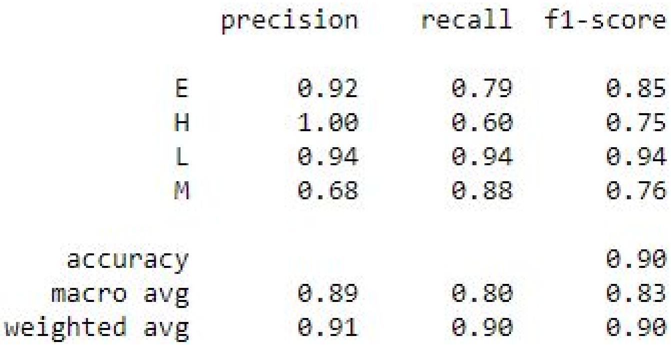


	0	1	2	3
0	22	0	7	0
1	0	2	2	1
2	5	0	134	4
3	1	5	13	5

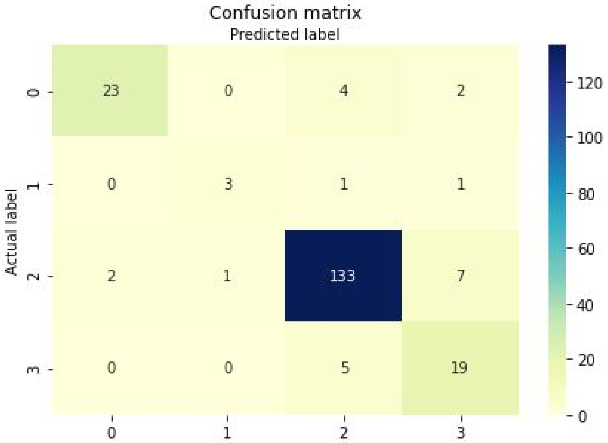
e) *Criterion*: Gini, max_depth: 11.
Using the parameters specified above, the average accuracy after cross validation through running the model and the dataset partition 50 times was:

Accuracy: 0.90070

The evaluation of one of the runned models is:



```
              precision    recall  f1-score

           E       0.92      0.79      0.85
           H       1.00      0.60      0.75
           L       0.94      0.94      0.94
           M       0.68      0.88      0.76

    accuracy                           0.90
   macro avg       0.89      0.80      0.83
weighted avg       0.91      0.90      0.90
```



Confusion matrix

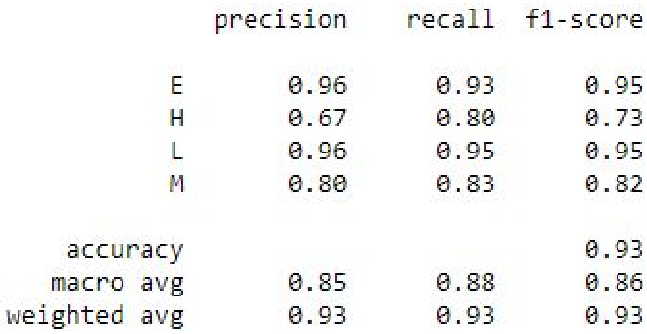
	0	1	2	3
0	23	0	4	2
1	0	3	1	1
2	2	1	133	7
3	0	0	5	19

f) *Criterion*: Entropy, max_depth: 11.

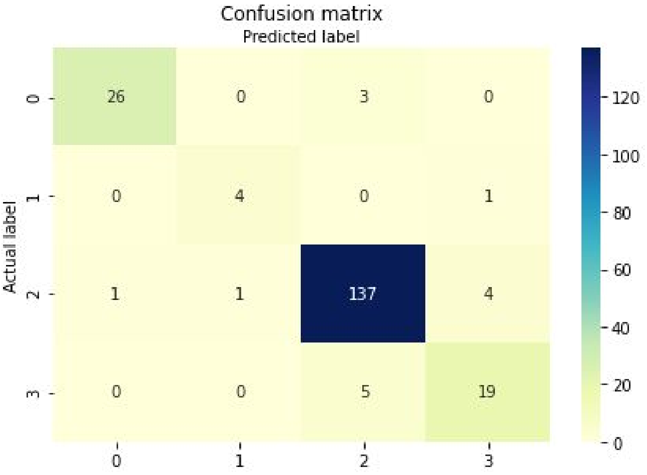
Using the parameters specified above, the average accuracy after cross validation through running the model and the dataset partition 50 times was:

Accuracy: **0.92368**

The evaluation of one of the runned models is:



```
              precision    recall  f1-score

           E       0.96      0.93      0.95
           H       0.67      0.80      0.73
           L       0.96      0.95      0.95
           M       0.80      0.83      0.82

    accuracy                           0.93
   macro avg       0.85      0.88      0.86
weighted avg       0.93      0.93      0.93
```



Confusion matrix

	0	1	2	3
0	26	0	3	0
1	0	4	0	1
2	1	1	137	4
3	0	0	5	19

Analysis.

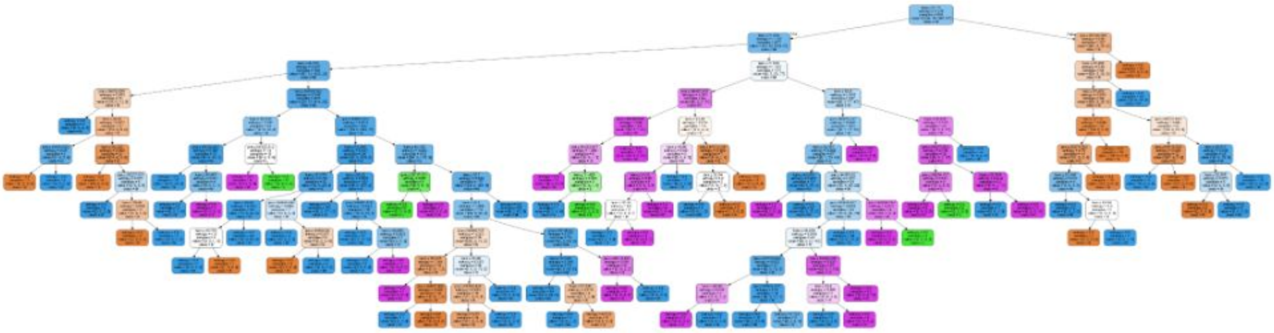
The following table summarizes the accuracy obtained from different Decision Tree models:

	Max_depth					
	None selected	*3*	*5*	*10*	***11***	*12*
Gini	0.89602	0.77114	0.82587	0.89635	0.90070	0.89881

Entropy	0.91857	0.76617	0.81094	0.92338	**0.92368**	0.91711

As it can be seen, the best model obtained was the one with the parameters “criterion = entropy” and “max_depth=11”. When the depth was lower than 11, the accuracy was also lower, so the model might have been underfitted. In a similar way, increasing the maximum depth did not result in a better performance since the model then was overfitted to the data. In general, the best criterion was “entropy” for almost all of the depths. Although the model’s depth might be high, reducing it to 3 or even 5 decreased significantly the model’s performance: it decreased approximately 0.16 and 0.1 respectively. It is also worth noting that each model was runned 50 times, so the average accuracies depicted in the table have a small variance.

The decision tree of the best model (entropy, max_depth=11) is the following:



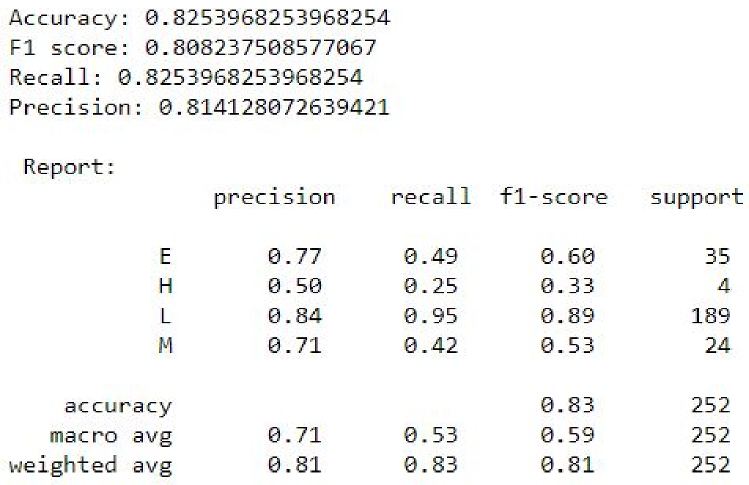
Since it is a big decision tree, it cannot be fully appreciated in detail, so the full image will be available in the folder with all the documentation.

2.2. SVM

For the SVM models, we used 2 different approaches:

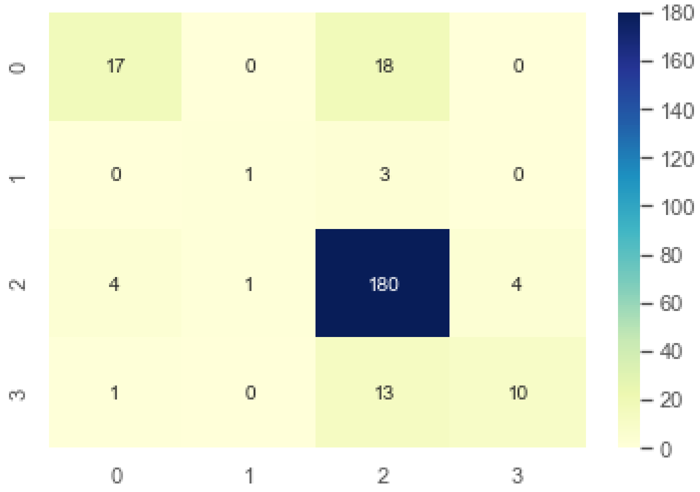
Polynomial SVM

First we trained a polynomial SVM, of grade 8, where we also divided the data into a 75% of training and 25% of testing. Getting the next results:



```
Accuracy: 0.8253968253968254
F1 score: 0.808237508577067
Recall: 0.8253968253968254
Precision: 0.814128072639421

 Report:
               precision    recall  f1-score   support

           E       0.77      0.49      0.60        35
           H       0.50      0.25      0.33         4
           L       0.84      0.95      0.89       189
           M       0.71      0.42      0.53        24

    accuracy                           0.83       252
   macro avg       0.71      0.53      0.59       252
weighted avg       0.81      0.83      0.81       252
```

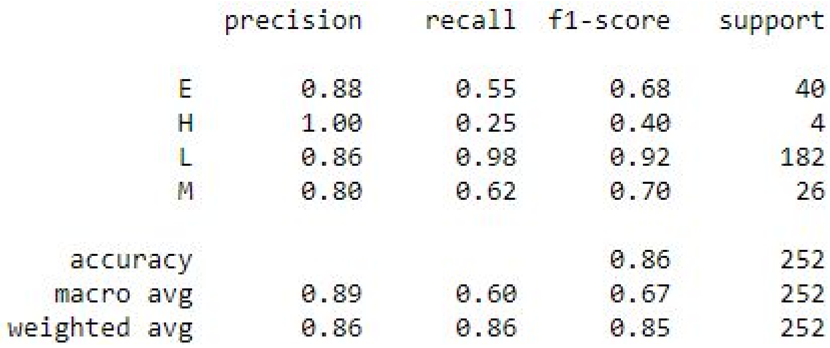
And the following confusion matrix:



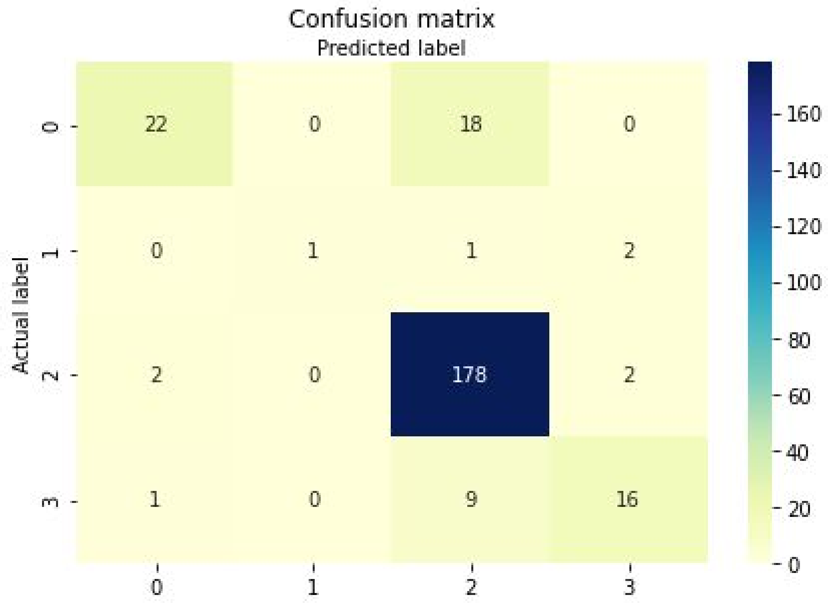
SVC Radial Basis Function (RBF) kernel

In this model we also split the dataset 75% for training and 25% for testing. The parameter used was *Gamma=0.01*. The results and confusion matrix obtained were:

Accuracy: 0.86111



```
              precision    recall  f1-score   support

           E       0.88      0.55      0.68        40
           H       1.00      0.25      0.40         4
           L       0.86      0.98      0.92       182
           M       0.80      0.62      0.70        26

    accuracy                           0.86       252
   macro avg       0.89      0.60      0.67       252
weighted avg       0.86      0.86      0.85       252
```

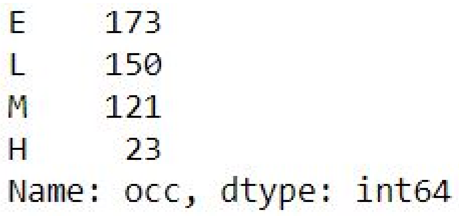


2.3. Neural Networks

For the neural network approach, we built 2 different configurations, the first one taking [tem, hum, pre] and other taking the dates in consideration.

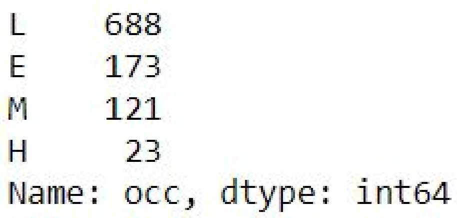
First Approach:

First, we saw that we had much more data in the class L, so we reduced that class to just 150 registers, getting:



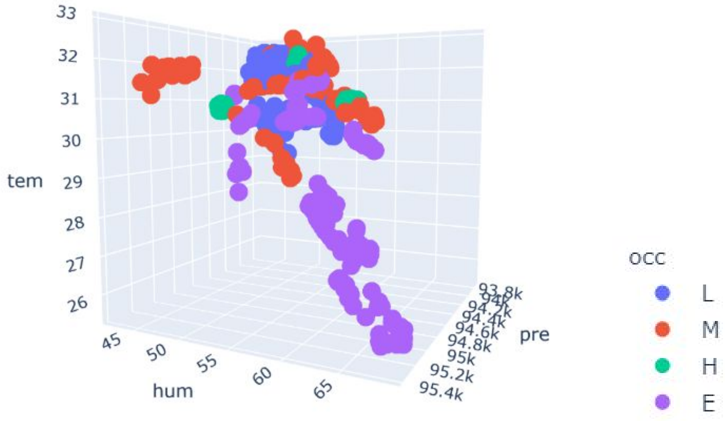
```
E    173
L    150
M    121
H     23
Name: occ, dtype: int64
```

instead of



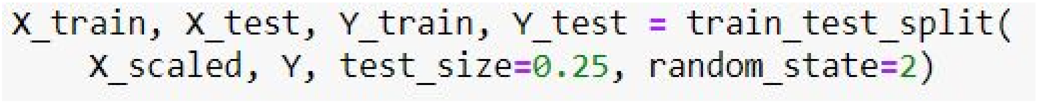
```
L    688
E    173
M    121
H     23
Name: occ, dtype: int64
```

This with the goal of avoiding a high accuracy and poor recall, because if the classifier just says everything is on the L class, it would still be correct most of the time.
plotting the data in a 3d scatter plot we get:



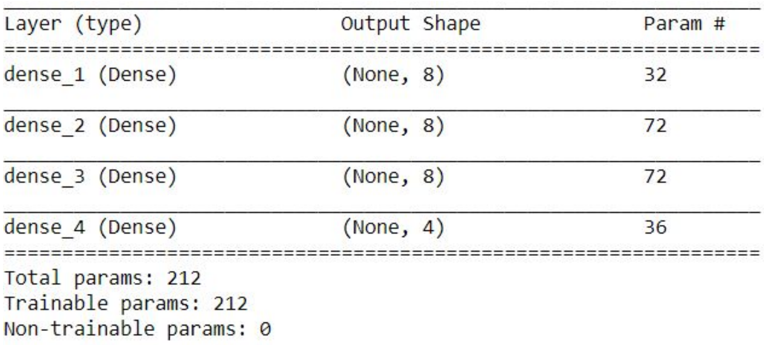
Where we can see that there is not a clear difference between all the classes.
We could think that the feature of temperature could divide some classes when this has a small value, but when the value goes higher, there is no linear way to divide the classes.

We divided this dataset with 75% train set and 25% test set, with a random state.



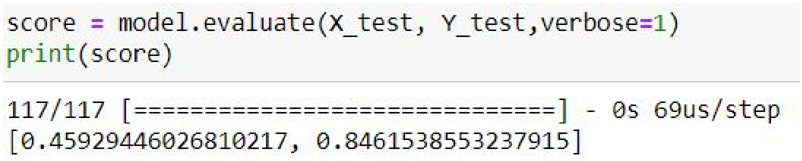
```
X_train, X_test, Y_train, Y_test = train_test_split(
    X_scaled, Y, test_size=0.25, random_state=2)
```

The model we trained has the next configuration:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 8)                 32        
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 72        
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 72        
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 36        
=================================================================
Total params: 212
Trainable params: 212
Non-trainable params: 0
```

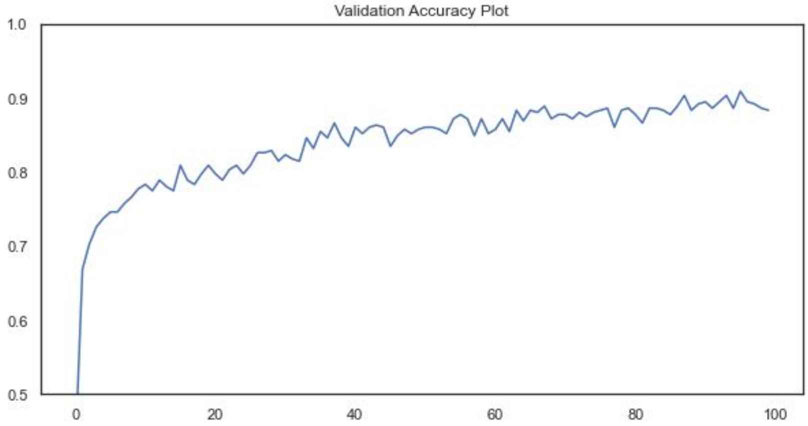
With 100 epochs getting an accuracy of 84%

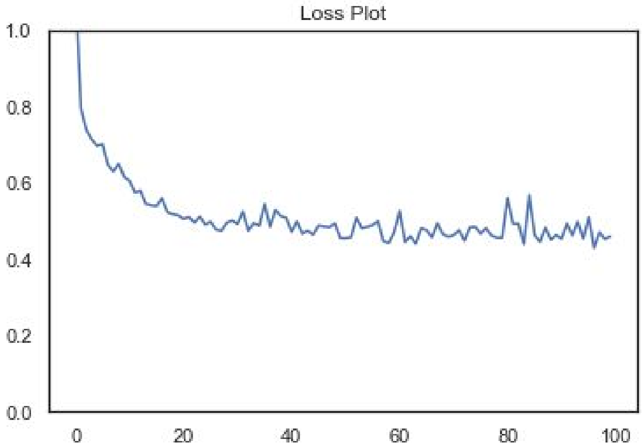


```
score = model.evaluate(X_test, Y_test,verbose=1)
print(score)
```

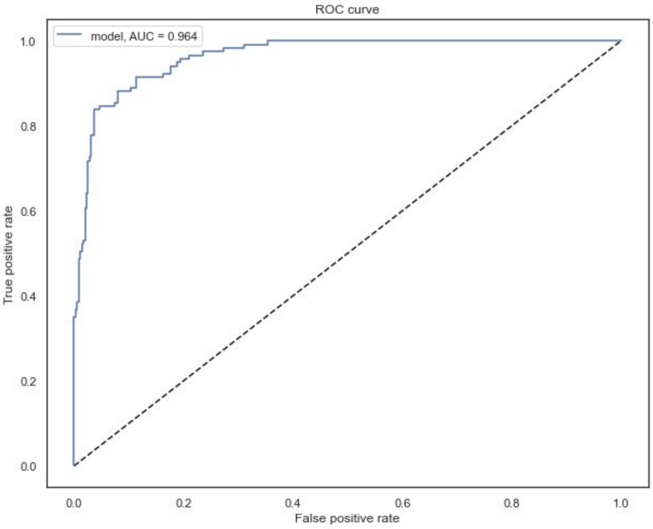
```
117/117 [==============================] - 0s 69us/step
[0.45929446026810217, 0.8461538553237915]
```

And the following validation accuracy and loss plots

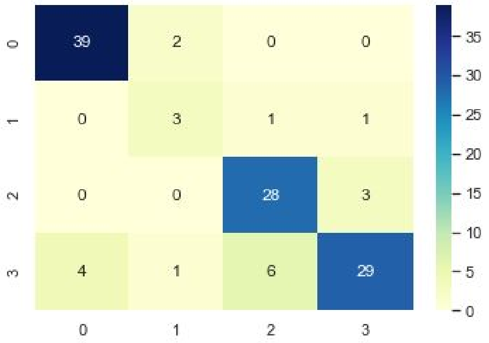




The following ROC curve, giving an AUC of 0.946



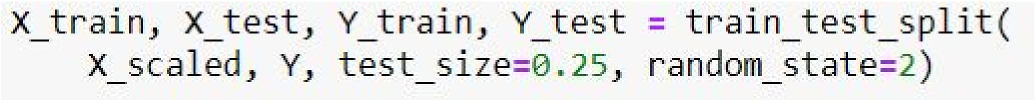
And the following confusion matrix



Second Approach:

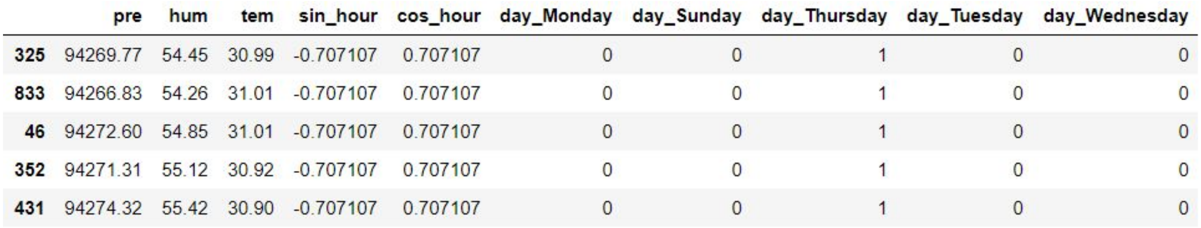
For the second approach, we generated more features to get better predictions, we thought about the date column, but not as it is, the best idea we had was to get the day of week of the date, and the hour of the register.
But for this approach, in order to take the hour feature, and not convert it into a categorical value, and dealing with the cyclic nature of the feature, we used the sine and cosine of the hour feature. (To handle the cycle of 23 -> 0 -> 1).

As the previous approach, we divided this dataset with 75% train set and 25% test set, with a random state.



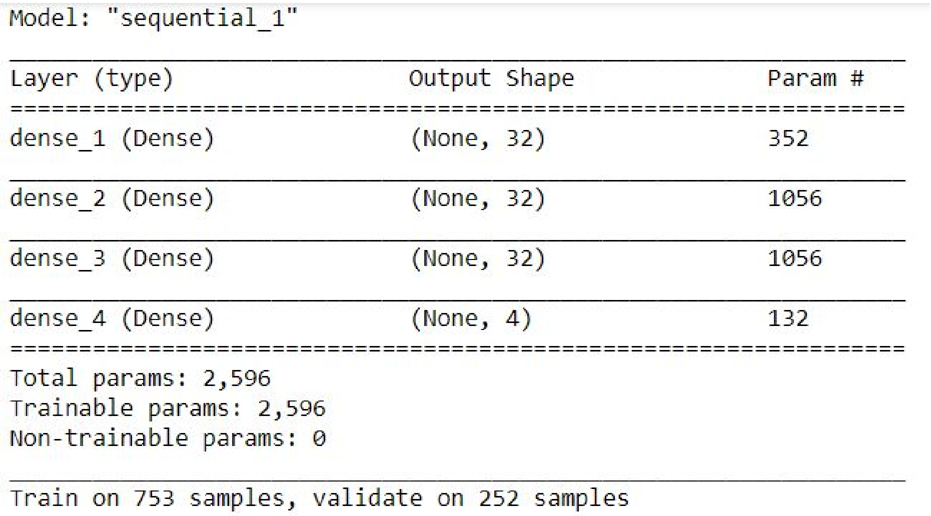
```
X_train, X_test, Y_train, Y_test = train_test_split(
    X_scaled, Y, test_size=0.25, random_state=2)
```

And getting the days of the week as a one hot encoded value.
Getting as the input data:



	pre	hum	tem	sin_hour	cos_hour	day_Monday	day_Sunday	day_Thursday	day_Tuesday	day_Wednesday
325	94269.77	54.45	30.99	-0.707107	0.707107	0	0	1	0	0
833	94266.83	54.26	31.01	-0.707107	0.707107	0	0	1	0	0
46	94272.60	54.85	31.01	-0.707107	0.707107	0	0	1	0	0
352	94271.31	55.12	30.92	-0.707107	0.707107	0	0	1	0	0
431	94274.32	55.42	30.90	-0.707107	0.707107	0	0	1	0	0

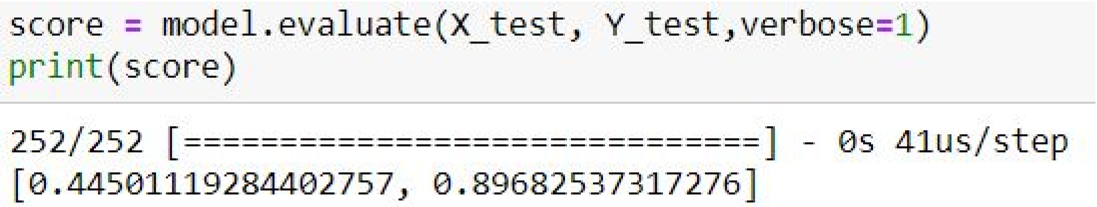
And having the next configuration:



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 32)                352       
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056      
_________________________________________________________________
dense_3 (Dense)              (None, 32)                1056      
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 132       
=================================================================
Total params: 2,596
Trainable params: 2,596
Non-trainable params: 0
_________________________________________________________________
Train on 753 samples, validate on 252 samples
```

Also noticing that in this case we are using all the data from all classes, as we are having more data to avoid the possible overfitting of the model.

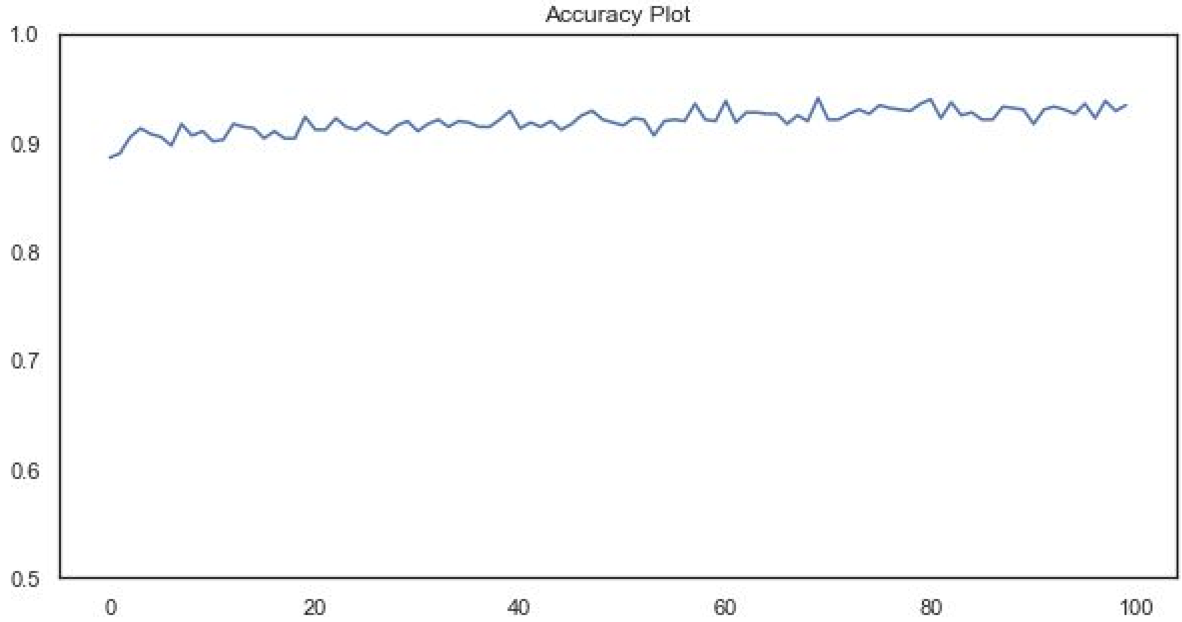
After training the model with 100 epochs we are getting a score of about 90%

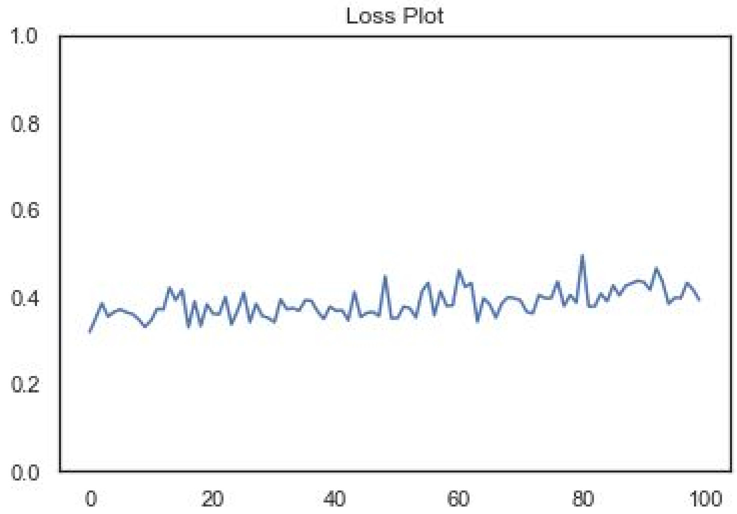


```
score = model.evaluate(X_test, Y_test,verbose=1)
print(score)
```

```
252/252 [==============================] - 0s 41us/step
[0.44501119284402757, 0.8968253731727 6]
```

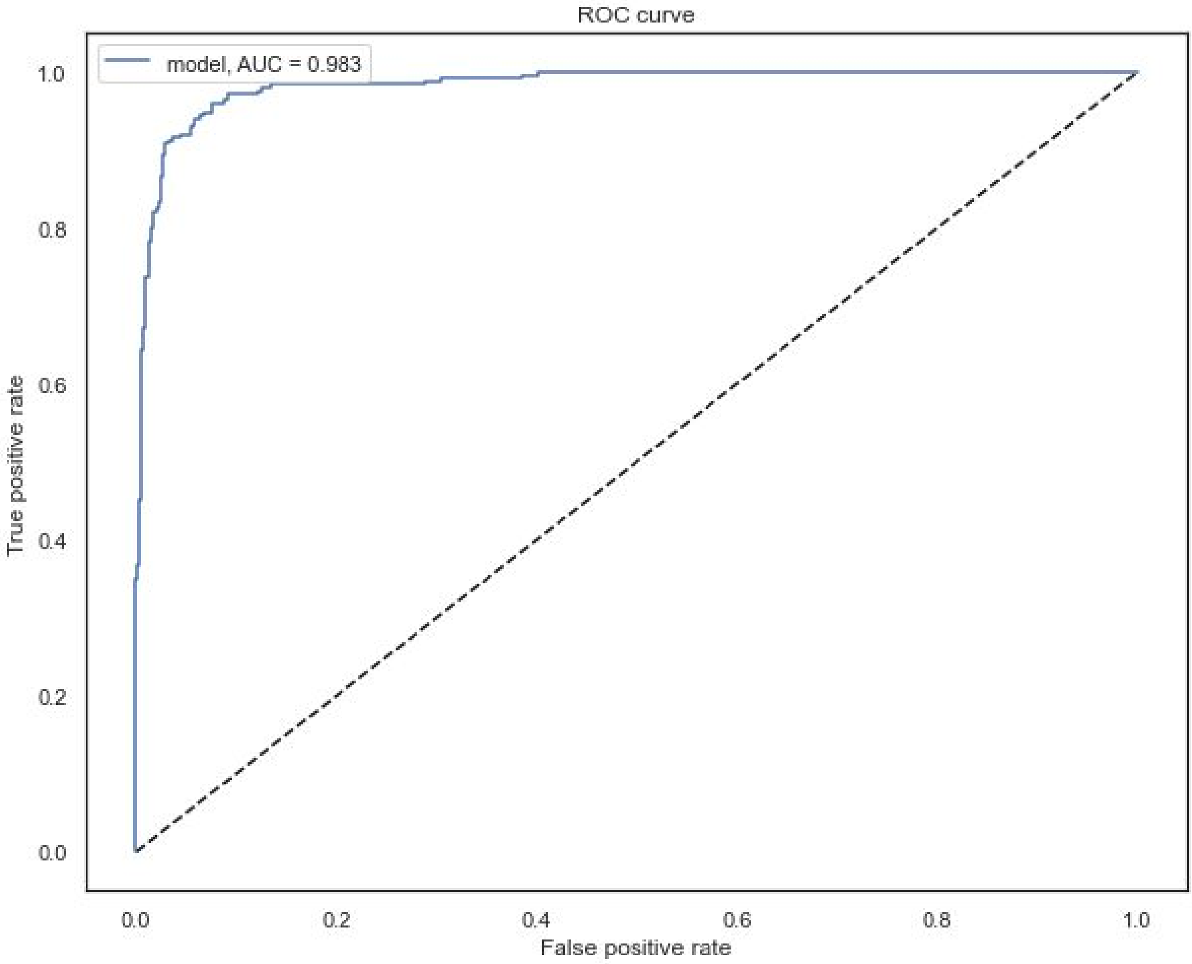
With the validation accuracy and loss plots:



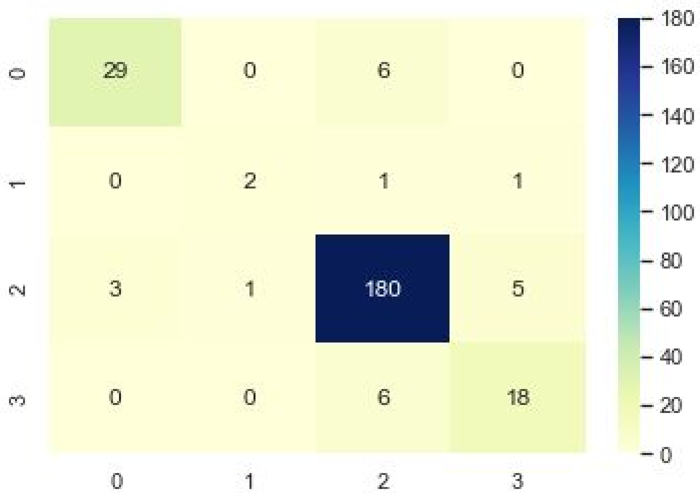


We notice that the accuracy plot is better than the previous model one, but the loss plot still doesn't look as we would want to, but the values are lower also.

We got the next ROC curve, getting an AUC of .983



And the next confusion matrix



	0	1	2	3
0	29	0	6	0
1	0	2	1	1
2	3	1	180	5
3	0	0	6	18

Where we can see directly that the second model has better accuracy, and also analyzing that for future data, the second model could generalize better, because of the more input features that we think are important.

3. Evaluation

The following table summarizes the characteristics of the best model of each category:

	Accuracy	Precision	Recall
Decision Trees	0.92368	E:0.96 H: 0.67 L:0.96 M: 0.80	E:0.93 H: 0.80 L:0.95 M: 0.83
SVM	0.86111	E:0.88 H: 1.00 L:0.86 M: 0.80	E:0.55 H: 0.25 L:0.98 M: 0.62
Neural Networks	0.9087	0.9142	0.9118

a) *Which was the best model and which metric did you use for choosing it?*

We choose the best model to be Neural Networks. This is because it has a high accuracy, and although it is smaller by 2% than the accuracy of the Decision Tree model, it has a higher global precision and recall. Also, the NN model uses 4 features (includes the time variable), whereas the Decision Tree model uses only 3 variables. Another important reason for why we chose NN is that Decision Trees are biased with imbalanced datasets, which was the case in this data since the occupation level was heavily loaded towards ‘L’.

b) *Which kind of errors were the most common for the chosen classifier?*

One of the most common mistakes of the NN classifier was categorizing as an ‘L’ instead of an ‘M’. In the confusion matrix shown in the section before, it can be observed that the model correctly classified 18 ‘M’, but classified incorrectly 6, and 5 out of these 6 were classified instead as ‘L’.

Another common mistake is not classifying correctly as ‘L’, and instead labeling as ‘M’ or ‘H’. This happened in 13 out of 193 values of L, so proportionally it is a less common mistake than the one described above.

c) *Which other characteristics could be generated from current data in order to improve your model?*

One thing we think helped in the predictions in the neural network, is to get the day of week of the date as a categorical variable, because since it is occupancy of a place, the actual day of week may be a critical value, like if the model could be modeled by a time series.