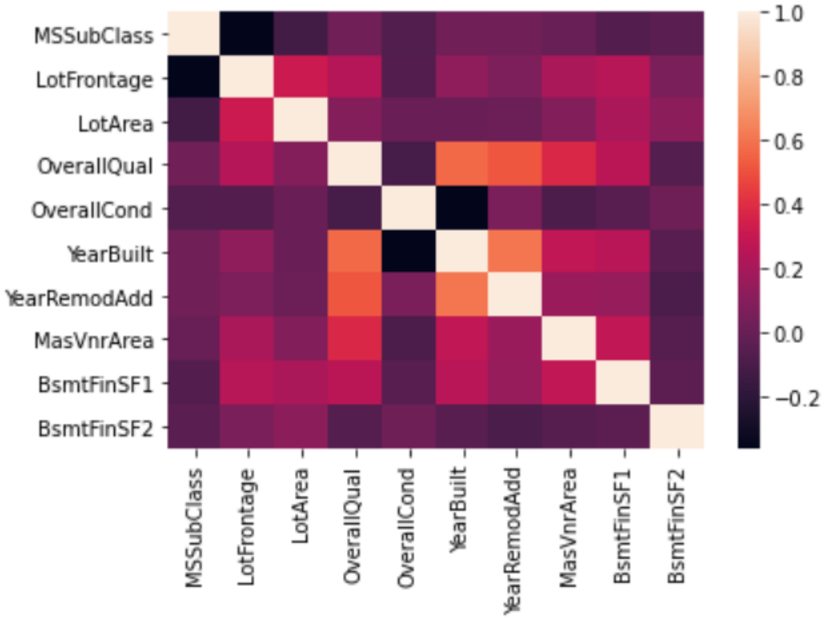
Q1

First we drop all the duplicates from test and training datasets, then we fill the blanks by finding it using fillna method. Fill in with the median of the column for numerical value and “None” for non numerical value. After that we combine the test and train datasets to get equal amount of feature columns using get_dummies funcction, then we separate it again for test and train datasets. Then we scale all the feature columns using StandardScaler and get the scaling factor from the training datasets. Then apply the scaling factor to the testing dataset.

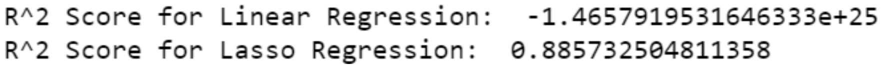
Q2

We visualize the first 10 feature columns relationships / correlations with each other on a heatmap as follows



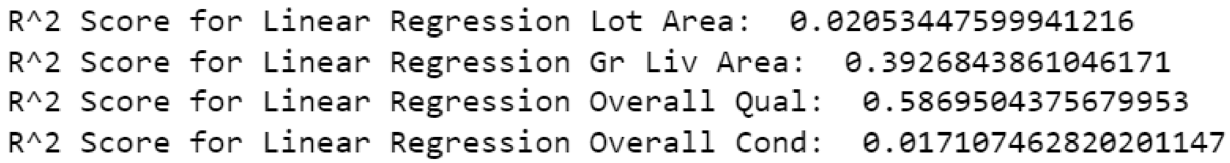
Q3

Here we try to select the features with the greatest weight towards the ground truth, i.e. the saleprice and label. This will also helps with computational load. We use the method SelectFromModel to help with selecting the more important features of the model. We first find the selector factor by using the training data to get the important feature columns, then apply it to the test data. Then we calculate how the model predict the test data using R square score. It is clear from the R square value that the LASSO model has a much better score and performs much better compared to the linear regression. This is probably due to the graph being too scattered with outliers and the model is trying to force a linear regression on the model where it’s not very compatible this time due to overfitting.



```
R^2 Score for Linear Regression:  -1.4657919531646333e+25
R^2 Score for Lasso Regression:  0.885732504811358
```

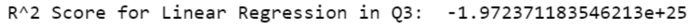
Q4



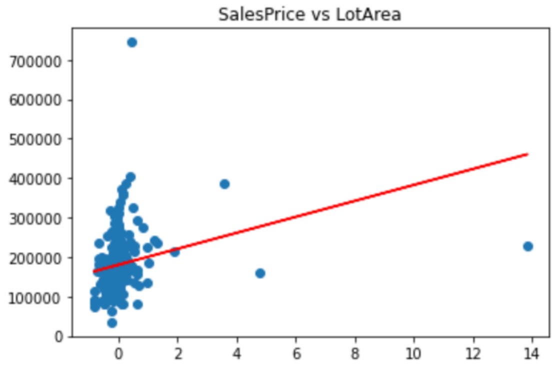
```
R^2 Score for Linear Regression Lot Area:  0.020534475999412l6
R^2 Score for Linear Regression Gr Liv Area:  0.3926843861046171
R^2 Score for Linear Regression Overall Qual:  0.5869504375679953
R^2 Score for Linear Regression Overall Cond:  0.01710746282020ll47
```

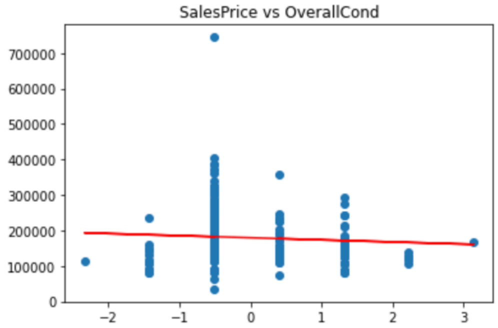
The R square score for each of the single feature is as follows. From the results, we can say that the Lot area and overall cond barely has any relationship with the saleprice, while the gr liv are and overall qual has a much stronger correlationship with saleprice.

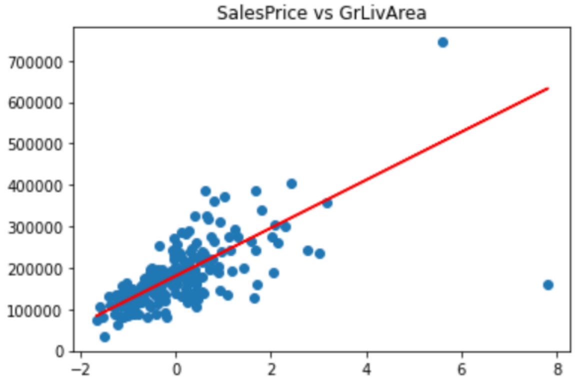
Q5

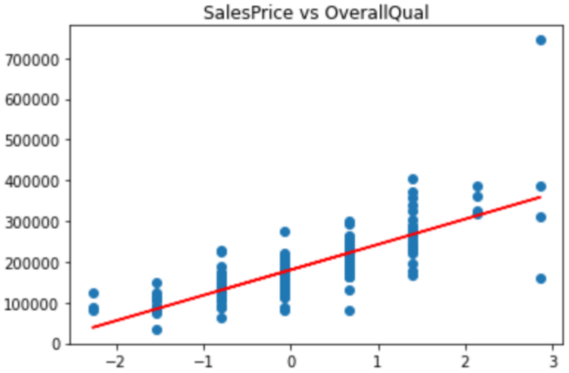


```
R^2 Score for Linear Regression in Q3:  -1.972371183546213e+25
```









The model is trained and used to predict the validation set, then plotted as the graph above.

Q6

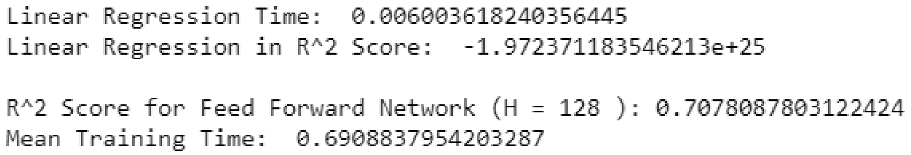
General model setting is 6 hidden layer of equal size (H), maximum iteration of 500, early stopping is set to true to prevent overfitting
R^2 Score for Feed Forward Network (H = 1 random_state= 12): -5.2266038743207
Training time: 0.046027421951293945 s
R^2 Score for Feed Forward Network (H = 1 random_state= 102): -5.226595903059875
Training time: 0.0490109920501709 s
R^2 Score for Feed Forward Network (H = 1 random_state= 3006): -5.226473283347459
Training time: 0.0490107536315918 s
Mean Training Time: 0.04801638921101888
STD Training Time: 0.0014064122402776369

```
Mean R^2 Score:  -5.226557686909345
STD R^2 Score:  5.9770986239538704e-05

R^2 Score for Feed Forward Network (H = 2 random_state= 12 ):  -5.2265352964441725
Training time:  0.06399369239807129 s
R^2 Score for Feed Forward Network (H = 2 random_state= 102 ):  -5.226459418432854
Training time:  0.06601786613464355 s
R^2 Score for Feed Forward Network (H = 2 random_state= 3006 ):  -5.226507976633178
Training time:  0.0620112419128418 s
Mean Training Time:  0.06400760014851888
STD Training Time:  0.0016357270519708803
Mean R^2 Score:  -5.226500897170068
STD R^2 Score:  3.137894462700218e-05

R^2 Score for Feed Forward Network (H = 4 random_state= 12 ):  -5.226643805047769
Training time:  0.06601667404174805 s
R^2 Score for Feed Forward Network (H = 4 random_state= 102 ):  -5.226418931812468
Training time:  0.07201457023620605 s
R^2 Score for Feed Forward Network (H = 4 random_state= 3006 ):  -5.226496923995242
Training time:  0.0710151195526123 s
Mean Training Time:  0.06968212127685547
STD Training Time:  0.002623782637370645
Mean R^2 Score:  -5.226519886951826
STD R^2 Score:  9.32289869801648e-05

R^2 Score for Feed Forward Network (H = 8 random_state= 12 ):  -5.226521054881458
Training time:  0.0760188102722168 s
R^2 Score for Feed Forward Network (H = 8 random_state= 102 ):  -5.226163285101222
Training time:  0.0710134506225586 s
R^2 Score for Feed Forward Network (H = 8 random_state= 3006 ):  -5.22642951074493
Training time:  0.07201647758483887 s
Mean Training Time:  0.07301624615987141
STD Training Time:  0.002162261040103405
Mean R^2 Score:  -5.22637128357587
STD R^2 Score:  0.00015175112591560075

R^2 Score for Feed Forward Network (H = 16 random_state= 12 ):  -5.226309225671451
Training time:  0.09302210807800293 s
R^2 Score for Feed Forward Network (H = 16 random_state= 102 ):  -5.2262507015434565
Training time:  0.08803081512451172 s
R^2 Score for Feed Forward Network (H = 16 random_state= 3006 ):  -5.226404443389288
Training time:  0.09901189804077148 s
Mean Training Time:  0.09335494041442871
STD Training Time:  0.004489181701718849
Mean R^2 Score:  -5.226321456868065
STD R^2 Score:  6.335792724852346e-05
```

```
R^2 Score for Feed Forward Network (H = 32 random_state= 12 ):  0.6396803827462656
Training time:  2.617586374282837 s
R^2 Score for Feed Forward Network (H = 32 random_state= 102 ):  0.6168047946637064
Training time:  3.8878746032714844 s
R^2 Score for Feed Forward Network (H = 32 random_state= 3006 ):  0.6591016466801733
Training time:  2.791623592376709 s
Mean Training Time:  3.0990281899770102
STD Training Time:  0.5623055137237116
Mean R^2 Score:  0.6385289413633818
STD R^2 Score:  0.017286802009421114

R^2 Score for Feed Forward Network (H = 64 random_state= 12 ):  0.5801764533093501
Training time:  0.4451000690460205 s
R^2 Score for Feed Forward Network (H = 64 random_state= 102 ):  0.641017638684608
Training time:  2.7766382694244385 s
R^2 Score for Feed Forward Network (H = 64 random_state= 3006 ):  0.6103726811810233
Training time:  0.8231837749481201 s
Mean Training Time:  1.3483073711395264
STD Training Time:  1.021708924111528
Mean R^2 Score:  0.6105222577249938
STD R^2 Score:  0.02483853510607399

R^2 Score for Feed Forward Network (H = 128 random_state= 12 ):  0.7039979712049755
Training time:  1.2732884883880615 s
R^2 Score for Feed Forward Network (H = 128 random_state= 102 ):  0.7123999945414186
Training time:  0.9802212715148926 s
R^2 Score for Feed Forward Network (H = 128 random_state= 3006 ):  0.707028375190333
Training time:  0.8271851539611816 s
Mean Training Time:  1.0268983046213787
STD Training Time:  0.18508755641842078
Mean R^2 Score:  0.7078087803122424
STD R^2 Score:  0.0034742167608695193
```

The overall mean R^2 is always -5.22 up to H=16, then it suddenly rise to 0.63 on H=32 and it tend to have an upward trend from there onwards.

Q7



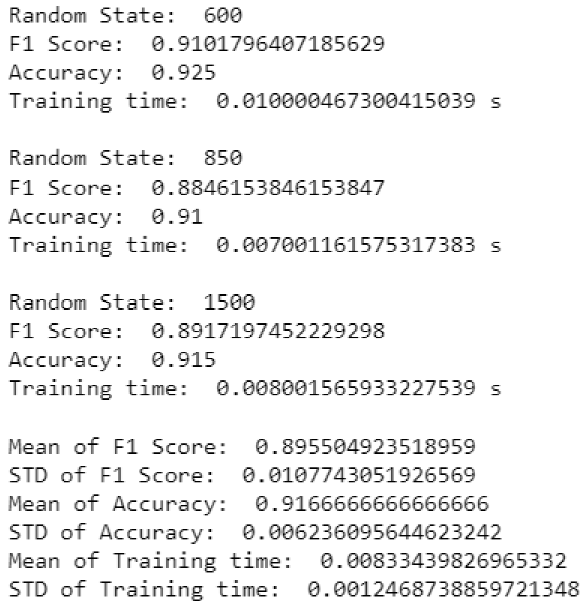
```
Linear Regression Time:  0.006003618240356445
Linear Regression in R^2 Score:  -1.972371183546213e+25

R^2 Score for Feed Forward Network (H = 128 ): 0.7078087803122424
Mean Training Time:  0.6908837954203287
```

Because Linear Regression don’t really work, it’s better to use neural network with 128 hidden units per layer.

Q8

In general, the trend is increasing when the hidden units also increases. This might be because there is around 93 selected feature columns selected, hence each unit can handle and learn one fo the feature columns, causing a better accuracy when the hidden units is higher, yet it might also cause overfitting if the hidden units is far above the selected feature columns.

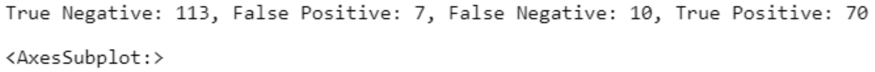
Q9



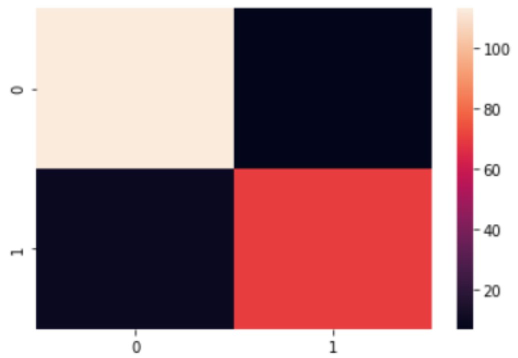
```
Random State:  600
F1 Score:  0.9101796407185629
Accuracy:  0.925
Training time:  0.010000467300415039 s

Random State:  850
F1 Score:  0.8846153846153847
Accuracy:  0.91
Training time:  0.007001161575317383 s

Random State:  1500
F1 Score:  0.8917197452229298
Accuracy:  0.915
Training time:  0.008001565933227539 s

Mean of F1 Score:  0.895504923518959
STD of F1 Score:  0.0107743051926569
Mean of Accuracy:  0.9166666666666666
STD of Accuracy:  0.006236095644623242
Mean of Training time:  0.00833439826965332
STD of Training time:  0.0012468738859721348
```

The general model setting is using make_pipeline and StandardScaler(), to help with scaling the features conveniently, then we use SGDClassifier with loss="log" method to use the logistic regression method on classifying the data. It is then followed by using a constant learning rate or step size of less than 1. Lastly, I use random state to get the mean data and std of some randomized state of the data in case the weight of the data were changed. The F1 score, accuracy and training time were as shown above.

Q10



```
True Negative: 113, False Positive: 7, False Negative: 10, True Positive: 70

<AxesSubplot:>
```



The values of confusion matrix were as shown above

One reason is because this confusion matrix provides insights to our predictions of the model. It is important for the model to know how precise and accurate it is for settings tuning later on to get a better model.

Q11

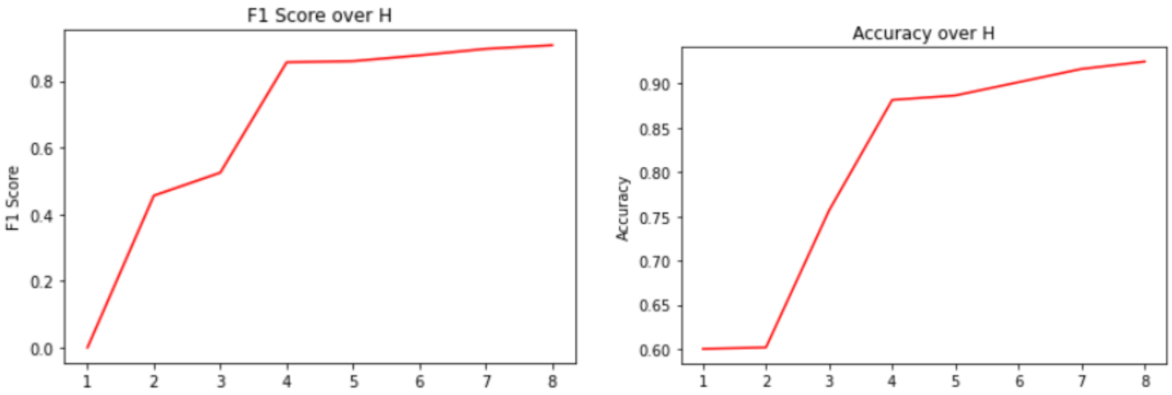
```
F1 Score      for H=  1 Random =, 600  :  0.0
Accuracy      for H=  1 Random =, 600  :  0.6
Training time for H=  1 Random =, 600  :  0.027014732360839844 s
F1 Score      for H=  1 Random =, 850  :  0.0
Accuracy      for H=  1 Random =, 850  :  0.6
Training time for H=  1 Random =, 850  :  0.028004884719848633 s
F1 Score      for H=  1 Random =, 1500  :  0.0
Accuracy      for H=  1 Random =, 1500  :  0.6
Training time for H=  1 Random =, 1500  :  0.029012441635131836 s

Mean of F1 Score      for H=  1  :  0.0
STD  of F1 Score      for H=  1  :  0.0
Mean of Accuracy      for H=  1  :  0.6
STD  of Accuracy      for H=  1  :  0.0
Mean of Training time for H=  1  :  0.02801068623860677
STD  of Training time for H=  1  :  0.0008155717133246613

F1 Score      for H=  2 Random =, 600  :  0.0
Accuracy      for H=  2 Random =, 600  :  0.6
Training time for H=  2 Random =, 600  :  0.04300522804260254 s
F1 Score      for H=  2 Random =, 850  :  0.7958115183246074
Accuracy      for H=  2 Random =, 850  :  0.805
Training time for H=  2 Random =, 850  :  0.2160487174987793 s
F1 Score      for H=  2 Random =, 1500  :  0.5714285714285715
Accuracy      for H=  2 Random =, 1500  :  0.4
Training time for H=  2 Random =, 1500  :  0.04401135444641113 s

Mean of F1 Score      for H=  2  :  0.455746696584393
STD  of F1 Score      for H=  2  :  0.33502807374917837
Mean of Accuracy      for H=  2  :  0.6016666666666667
STD  of Accuracy      for H=  2  :  0.1653447576698121
Mean of Training time for H=  2  :  0.10102176666259766
STD  of Training time for H=  2  :  0.0813373740932876

F1 Score      for H=  4 Random =, 600  :  0.7432432432432432
Accuracy      for H=  4 Random =, 600  :  0.81
Training time for H=  4 Random =, 600  :  0.20904898643493652 s
F1 Score      for H=  4 Random =, 850  :  0.8301886792452831
Accuracy      for H=  4 Random =, 850  :  0.865
Training time for H=  4 Random =, 850  :  0.17203783988952637 s
F1 Score      for H=  4 Random =, 1500  :  0.0
```

```
Accuracy      for H=  4 Random =, 1500  :  0.595
Training time for H=  4 Random =, 1500  :  0.04100918769836426 s

Mean of F1 Score      for H=  4  :  0.5244773074961754
STD  of F1 Score      for H=  4  :  0.37255622552774903
Mean of Accuracy      for H=  4  :  0.7566666666666667
STD  of Accuracy      for H=  4  :  0.11649988078200864
Mean of Training time for H=  4  :  0.14069867134094238
STD  of Training time for H=  4  :  0.0720923070507265

F1 Score      for H=  8 Random =, 600  :  0.8402366863905325
Accuracy      for H=  8 Random =, 600  :  0.865
Training time for H=  8 Random =, 600  :  0.13703083992004395 s
F1 Score      for H=  8 Random =, 850  :  0.8625
Accuracy      for H=  8 Random =, 850  :  0.89
Training time for H=  8 Random =, 850  :  0.18204164505004883 s
F1 Score      for H=  8 Random =, 1500  :  0.8641975308641976
Accuracy      for H=  8 Random =, 1500  :  0.89
Training time for H=  8 Random =, 1500  :  0.09001994132995605 s

Mean of F1 Score      for H=  8  :  0.8556447390849101
STD  of F1 Score      for H=  8  :  0.010917156792048684
Mean of Accuracy      for H=  8  :  0.8816666666666667
STD  of Accuracy      for H=  8  :  0.011785113019775804
Mean of Training time for H=  8  :  0.13636414210001627
STD  of Training time for H=  8  :  0.03757066101250244

F1 Score      for H=  16 Random =, 600  :  0.8701298701298702
Accuracy      for H=  16 Random =, 600  :  0.9
Training time for H=  16 Random =, 600  :  0.1400313377380371 s
F1 Score      for H=  16 Random =, 850  :  0.8658536585365854
Accuracy      for H=  16 Random =, 850  :  0.89
Training time for H=  16 Random =, 850  :  0.1450328826904297 s
F1 Score      for H=  16 Random =, 1500  :  0.8414634146341463
Accuracy      for H=  16 Random =, 1500  :  0.87
Training time for H=  16 Random =, 1500  :  0.11902523040771484 s

Mean of F1 Score      for H=  16  :  0.8591489811002005
STD  of F1 Score      for H=  16  :  0.012626848176710727
Mean of Accuracy      for H=  16  :  0.8866666666666667
STD  of Accuracy      for H=  16  :  0.012472191289246483
Mean of Training time for H=  16  :  0.13469648361206055
STD  of Training time for H=  16  :  0.011267800605375109

F1 Score      for H=  32 Random =, 600  :  0.9068322981366459
Accuracy      for H=  32 Random =, 600  :  0.925
```

```
Training time for H=  32 Random =, 600  :  0.16203761100769043 s
F1 Score      for H=  32 Random =, 850  :  0.888888888888889
Accuracy      for H=  32 Random =, 850  :  0.91
Training time for H=  32 Random =, 850  :  0.11304068565368652 s
F1 Score      for H=  32 Random =, 1500  :  0.8333333333333334
Accuracy      for H=  32 Random =, 1500  :  0.87
Training time for H=  32 Random =, 1500  :  0.08601951599121094 s

Mean of F1 Score      for H=  32  :  0.8763515067862895
STD  of F1 Score      for H=  32  :  0.03128805863029942
Mean of Accuracy      for H=  32  :  0.9016666666666667
STD  of Accuracy      for H=  32  :  0.023213980461973552
Mean of Training time for H=  32  :  0.12036593755086263
STD  of Training time for H=  32  :  0.03146354693929085

F1 Score      for H=  64 Random =, 600  :  0.8848484848484848
Accuracy      for H=  64 Random =, 600  :  0.905
Training time for H=  64 Random =, 600  :  0.18703866004943848 s
F1 Score      for H=  64 Random =, 850  :  0.9056603773584907
Accuracy      for H=  64 Random =, 850  :  0.925
Training time for H=  64 Random =, 850  :  0.19004249572753906 s
F1 Score      for H=  64 Random =, 1500  :  0.8974358974358975
Accuracy      for H=  64 Random =, 1500  :  0.92
Training time for H=  64 Random =, 1500  :  0.17103815078735352 s

Mean of F1 Score      for H=  64  :  0.8959815865476243
STD  of F1 Score      for H=  64  :  0.008558425968281106
Mean of Accuracy      for H=  64  :  0.9166666666666666
STD  of Accuracy      for H=  64  :  0.008498365855987983
Mean of Training time for H=  64  :  0.1827064355214437
STD  of Training time for H=  64  :  0.008341359148728253

F1 Score      for H=  128 Random =, 600  :  0.8944099378881988
Accuracy      for H=  128 Random =, 600  :  0.915
Training time for H=  128 Random =, 600  :  0.20904827117919922 s
F1 Score      for H=  128 Random =, 850  :  0.9146341463414636
Accuracy      for H=  128 Random =, 850  :  0.93
Training time for H=  128 Random =, 850  :  0.25806546211242676 s
F1 Score      for H=  128 Random =, 1500  :  0.9113924050632911
Accuracy      for H=  128 Random =, 1500  :  0.93
Training time for H=  128 Random =, 1500  :  0.3630802631378174 s

Mean of F1 Score      for H=  128  :  0.9068121630976512
STD  of F1 Score      for H=  128  :  0.008868995216976712
Mean of Accuracy      for H=  128  :  0.9250000000000002
STD  of Accuracy      for H=  128  :  0.007071067811865481
```

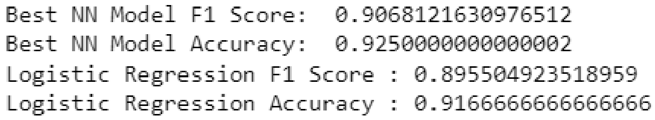
Mean of Training time for H= 128 : 0.27673133214314777
STD of Training time for H= 128 : 0.06425353253421313
The general setting for the model is using the MLPClassifier method with 3 hidden layer, max iteration of 500, early_stopping set to true to prevent overfitting and random_state based on the set values

Q12



Accuracy is used to calculate true negative and true positive, while F1 is used to calculate false positive and false negative. There is a gap between these two metrics because there is a gap in the confusion matrix, such that there exist a minority either it be the positive class or the negative class, hence there is a gap between the F1 score and accuracy score.

Q13



```
Best NN Model F1 Score:  0.9068121630976512
Best NN Model Accuracy:  0.9250000000000002
Logistic Regression F1 Score : 0.895504923518959
Logistic Regression Accuracy : 0.9166666666666666
```

From the results, the best neural network model has a better performance and results compared to the logistic regression.

Q14

In general, the trend is increasing when the hidden units also increases. This might be because there is around 93 selected feature columns selected, hence each unit can handle and learn one fo the feature columns, causing a better accuracy when the hidden units is higher, yet it might also cause overfitting if the hidden units is far above the selected feature columns. But since the feature columns and hidden units are still comparably near to each other hence why it's better.

Q15

Completed data in the jupyter notebook
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (32, 32, 32), 'learning_rate_init': 0.001, 'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (32, 32, 32), 'learning_rate_init': 0.01, 'max_iter': 500, 'random_state': 4211},

```
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (32, 32, 32), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.001,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.01,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init':
0.001, 'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init':
0.01, 'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (256, 256, 256), 'learning_rate_init':
0.001, 'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (256, 256, 256), 'learning_rate_init':
0.01, 'max_iter': 500, 'random_state': 4211},
{'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (256, 256, 256), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (32, 32, 32), 'learning_rate_init': 0.001,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (32, 32, 32), 'learning_rate_init': 0.01,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (32, 32, 32), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.001,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.01,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init': 0.001,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init': 0.01,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (256, 256, 256), 'learning_rate_init': 0.001,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (256, 256, 256), 'learning_rate_init': 0.01,
'max_iter': 500, 'random_state': 4211},
{'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (256, 256, 256), 'learning_rate_init': 0.1,
'max_iter': 500, 'random_state': 4211}]
```

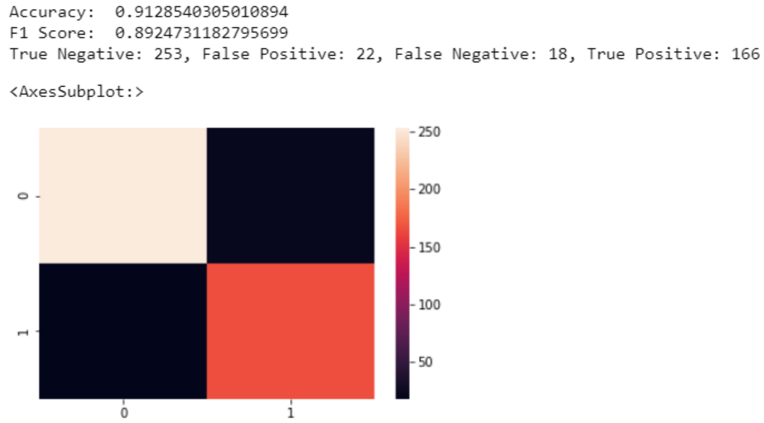
Q16

The top 3 of the settings from the grid search is
First one: {'alpha': 0.1, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init': 0.01, 'max_iter': 500, 'random_state': 4211}
Accuracy: 0.93
Mean score: 0.9337500000000001
STD score: 0.004999999999999982

Second one: {'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (128, 128, 128), 'learning_rate_init': 0.01, 'max_iter': 500, 'random_state': 4211}
Accuracy: 0.93
Mean score: 0.9325000000000001
STD score: 0.002499999999999991

Third one: {'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (64, 64, 64), 'learning_rate_init': 0.1, 'max_iter': 500, 'random_state': 4211}
Accuracy: 0.92
Mean score: 0.925
STD score: 0.007905694150420974

Q17



```
Accuracy:  0.9128540305010894
F1 Score:  0.8924731182795699
True Negative: 253, False Positive: 22, False Negative: 18, True Positive: 166

<AxesSubplot:>
```

The best parameter is shown in Q16, then we take that and train it using the whole training set, then test it using the test dataset to get the above accuracy and F1 score. The we get the confusion matrix as above too.