De La Salle University – Manila

Gokongwei College of Engineering

Electronics and Communications Engineering Department

Intelligent Systems for Engineering

LBYEC3B

Project 1 Documentation

Wine Quality Classification

by

BAYETA IV, Reginald Geoffrey L. – 11811269

TAN, Adrienne Rafaela P. – 11809736

TUPAL, Isaiah Jassen L. – 11847115

LBYEC3B – EQ4

Project 1: Wine Quality Classification

I. Statistical Analysis

Bias: Training data columns *wineType* and *id* were removed.

Reasons:

- Parameter *id* is just for identification or index for each data sample.
- For *wineType*, there has been a previous attempt to separate data (red and white wine) and used it for training neural networks but outputs low accuracy.
- In addition, oversampling was done to both red and white wine train data which outputs high accuracy prediction as per the models, but low on actual accuracy as seen in Figure 1.
- To produce the outputs, both training data are being fed to a fitnet() model with hidden layers of [9 5 5 2] then the combined predicted results for both types are exported as *.csv* file (Refer to Appendix A).

Figure 1. *Outputs for separated training parameters (by wineType) using fitnet*

From the aforementioned, we now have 12 parameters remaining: 11 features and 1 target (quality). To know more about the data, we generate a histogram and summary for each parameter as seen in Figure 2 and Table 1.

Table 1. *Wine Train Summary*

index	Parameters	Min	Median	Max
1	fixedAcidity	4.2	7.1	15.9
2	volatileAcidity	0.08	0.29	1.58
3	citricAcid	0	0.32	1.66
4	residualSugar	0.6	2.9	65.8
5	chlorides	0.012	0.047	0.611

6	freeSulfurDioxide	1	29	146.5
7	totalSulfurDioxide	6	122	366.5
8	density	0.98713	0.9952	1.039
9	pH	2.72	3.21	3.90
10	sulphates	0.22	0.51	2
11	alcohol	8	10.1	14.9
12	quality	3	6	9

By inspection from Table 1, it can be observed that there are outliers present in the data as seen in the disparity of the min and max of multiple table variables such as index 4, 6, 7, and 10. We can also see that there are parameters that fall within a small range such as in index 9, *pH*, as wines are usually on the acidic side of the spectrum of pH. We can confirm these observations by looking at Figure 2.

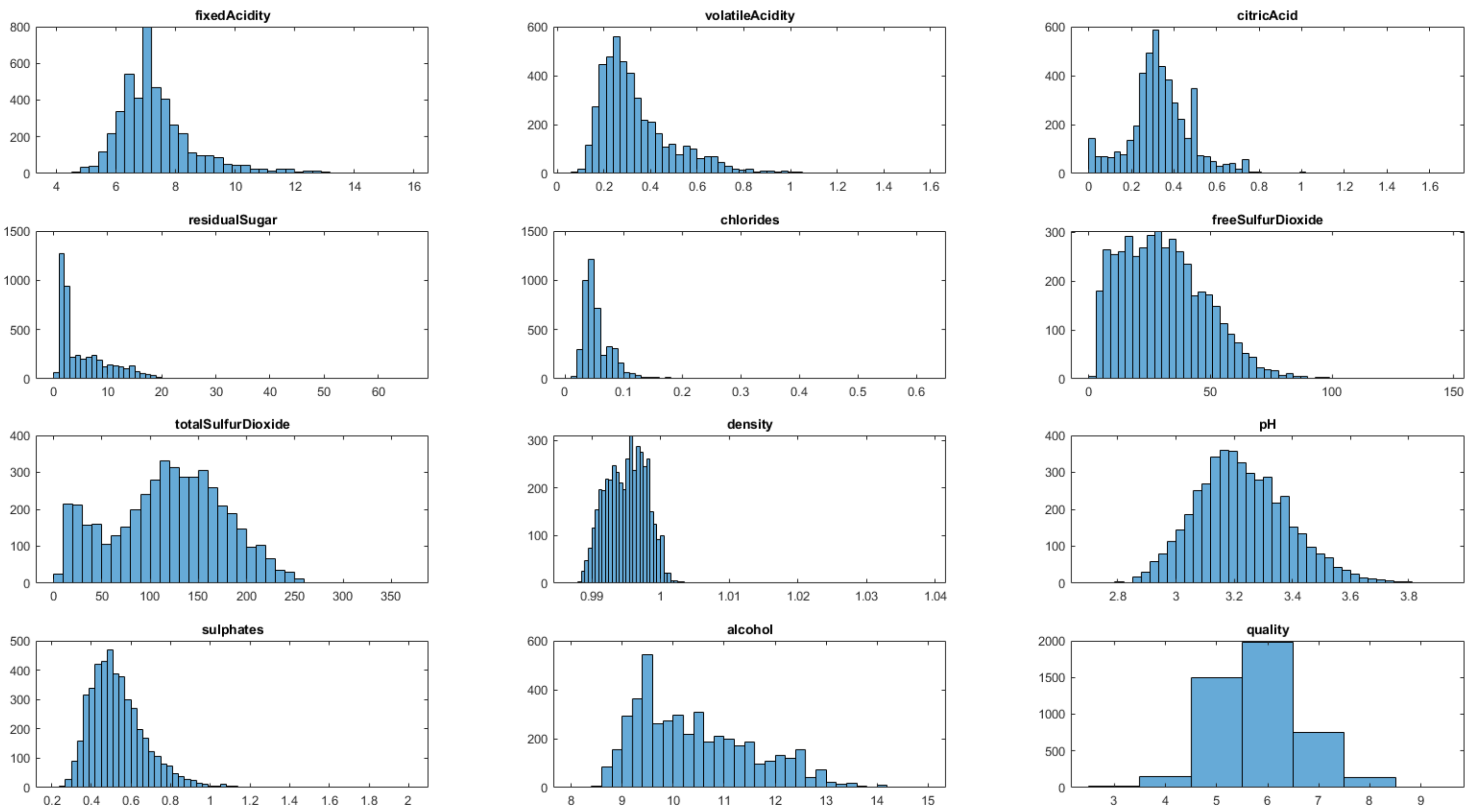


Figure 2. *Wine Train Data Histograms*

It should be evident that descriptive observations are subject to biases, and to combat this, we added another layer of proof for statistical observation: a correlation matrix (*corrplot*()) was generated that shows the correlations among pairs of variables in the training data as seen in Figure 3. In this correlation matrix, the row to focus would be *quality* vs other parameters which is summarized in Table 2. From Table 2, it is apparent that the parameters with outliers such as index 4, 6, 7, and 10 are found to be in the middle in the intuitive ranking, which denotes little to no correlation compared to the outer extremes (-/+).

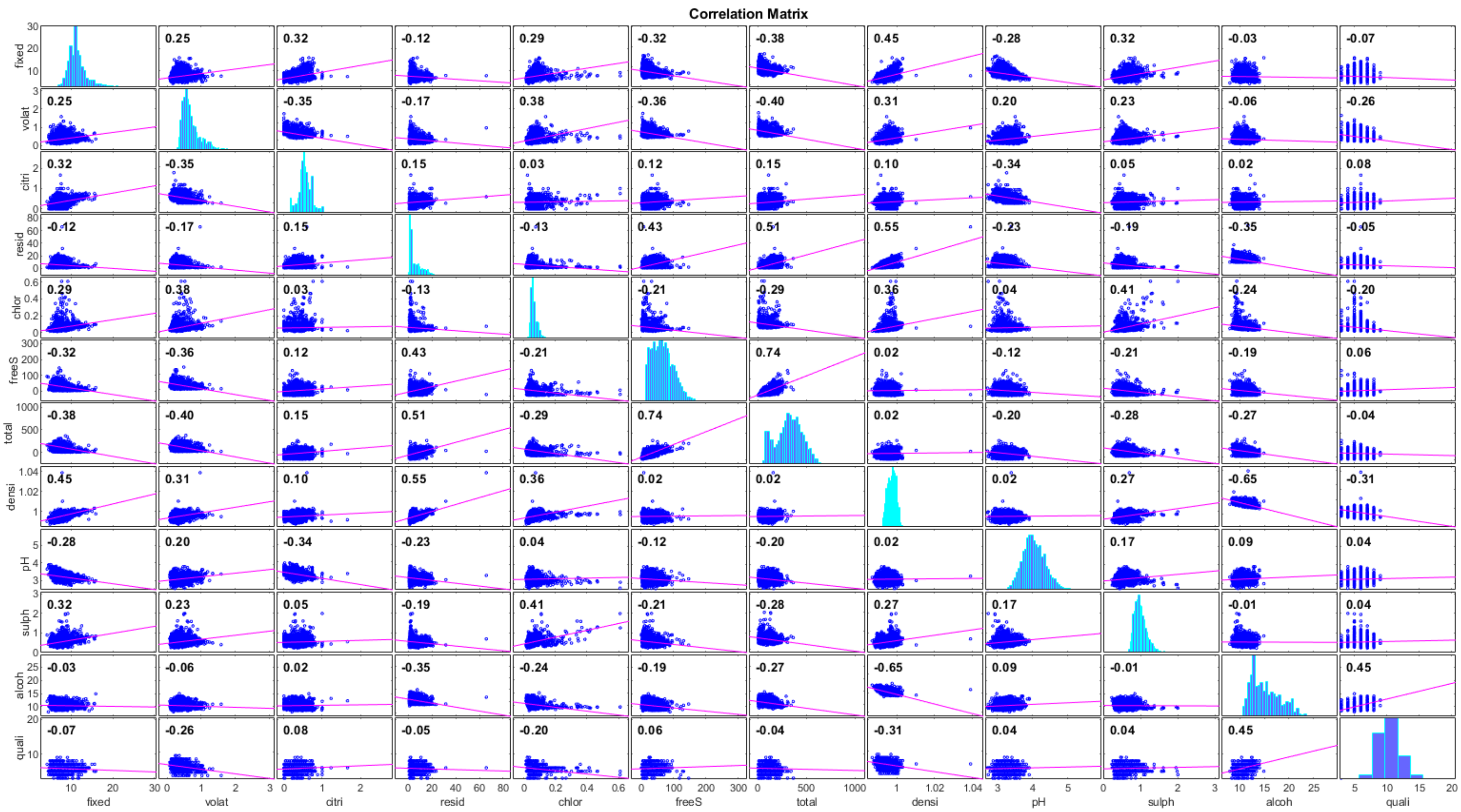


Figure 3. *Wine Train Correlation Matrix*

Table 2. *Wine Train Summary by Correlation Result (with respect to quality)*

Index	Parameters	Min	Median	Max	Correlation Score
8	density	0.98713	0.9952	1.039	-0.31
2	volatileAcidity	0.08	0.29	1.58	-0.26
5	chlorides	0.012	0.047	0.611	-0.20
1	fixedAcidity	4.2	7.1	15.9	-0.07
4	residualSugar	0.6	2.9	65.8	-0.05
7	totalSulfurDioxide	6	122	366.5	-0.04
9	pH	2.72	3.21	3.90	+0.04
10	sulphates	0.22	0.51	2	+0.04

6	freeSulfurDioxide	1	29	146.5	+0.06
3	citricAcid	0	0.32	1.66	+0.08
11	alcohol	8	10.1	14.9	+0.45

II. Preprocessing

The preprocessing step comprises of two parts: feature ranking and normalization of new training and testing data from selected ranked features. In addition, the results of the statistical analysis and feature ranking will be discussed.

a. Selecting feature ranking function

Upon checking MATLAB documentation, two main feature rankings were explored in this project: fscmrmr ("Rank features for classification using minimum redundancy maximum relevance (MRMR) algorithm") and fscchi2 ("Univariate feature ranking for classification using chi-square tests"). The latter was used as it performed better in previous submission attempts in terms of accuracy as seen in Figure 5, which outputs a high accuracy of 0.53976 compared to all samples shown in Figures 4 and 5.

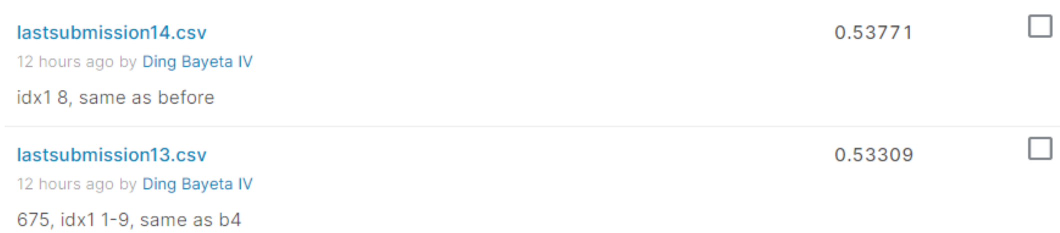


Figure 4. *Accuracy utilizing fscmrmr()*

NB. idx1 refers to the index list of the feature ranking generated by fscmrmr lastsubmission14 used Top 8 selected ranked features, while lastsubmission13 used Top 9.

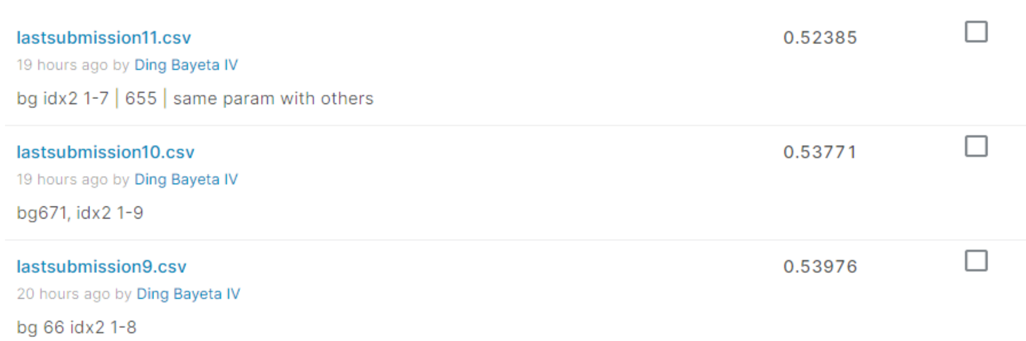


Figure 5. *Accuracy utilizing fscchi2()*

NB. idx2 refers to the index list of the feature ranking generated by fscchi2() lastsubmission11 → Top 7, lastsubmission9 → Top 8, lastsubmission10 → Top 9 selected ranked features.

b. Generating feature ranking of training data by using fscchi2

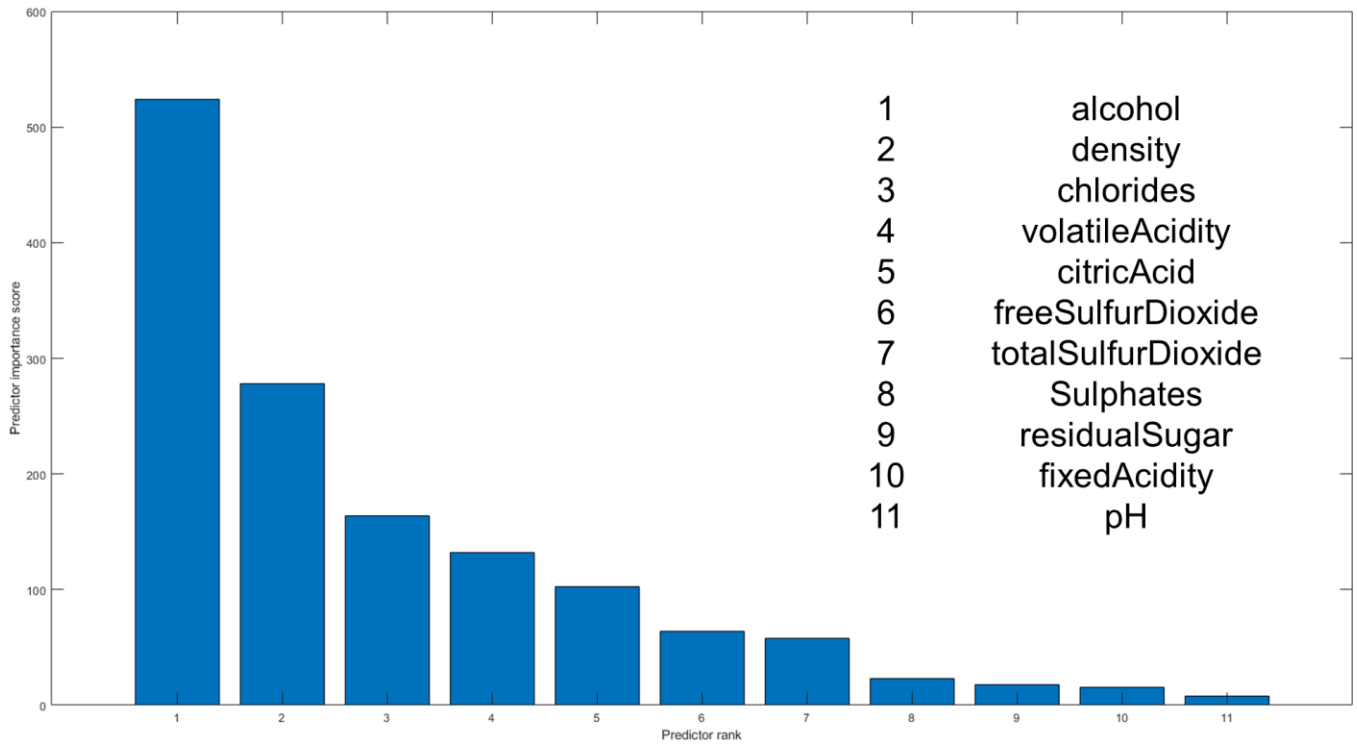


Figure 6. *Predictor rank vs Predictor importance score for fscchi2*

When calling fscchi2, it returns two parameters, *idx2* and *scores2* (variables used in code) which represent the column index sorted by top feature rankings and their corresponding scores. As per the documentation of fscchi2, a higher score indicates that its corresponding predictor variable is important relative to the target parameter. Table 3 highlights the top column features and corresponding scores, which is also reflected in Figure 6. In addition, the returned score matches with the correlation score as well as the discussed descriptive observations.

Table 3. *fscchi2 Top column features and corresponding scores*

Ranking	Column Index	Feature	Score	Correlation Score
1	11	alcohol	524.07	+0.45
2	8	density	278.25	-0.31
3	5	chlorides	163.76	-0.20
4	2	volatileAcidity	132.27	-0.26
5	3	citricAcid	102.38	+0.08
6	6	freeSulfurDioxide	63.745	+0.06
7	7	totalSulfurDioxide	57.681	-0.04
8	10	sulphates	23.13	+0.04

9	4	residualSugar	18.091	-0.05
10	1	fixedAcidity	15.259	-0.07
11	9	pH	7.7558	+0.04

c. Generating normalized new training and testing data from selected ranked features

The selected range for the top-ranked features to be used in the final training and testing data to be normalized is the Top 8 with the results shown in Figure 5 as the basis. Using Top 8 features outperforms Top 7 and Top 9 in terms of accuracy.

```
% Normalize train and test data
norm_train = normalize(wine_train);
norm_test  = normalize(wine_test);
% Select ranked features
mask = idx2(1:8)
wine_feature = norm_train{:, mask};
wine_target  = double(qlty);
wine_testing = norm_test{:, mask};
```

III. Models and Training

Figure 7. *Classification Learner models used for training and testing*

IV. Testing Results

a. Accuracy of models in validation vs official submission

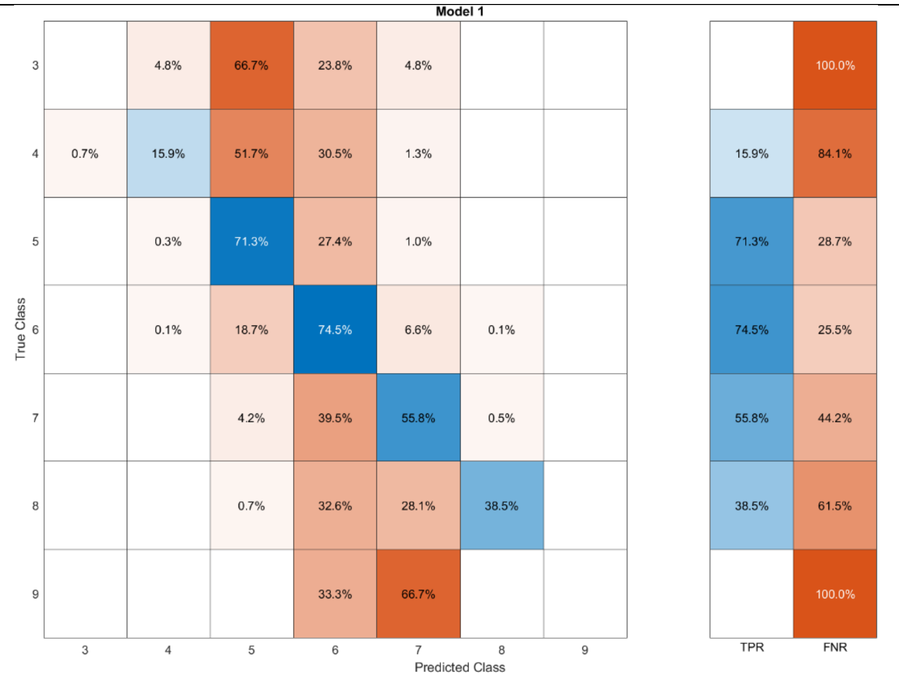
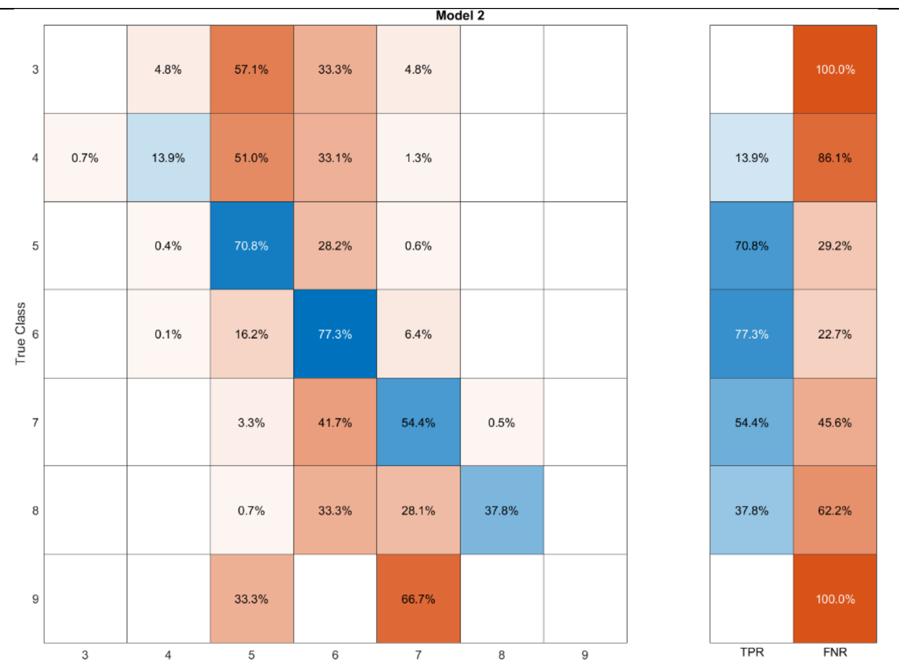
As shown in Table 4, Bagged Trees, or also referred to as the Random Forest model has garnered the best accuracy out of 17 instances of models trained. Bagged Trees with 500 learners outclassed the other Bagged Trees instances with a 0.56695 accuracy score.

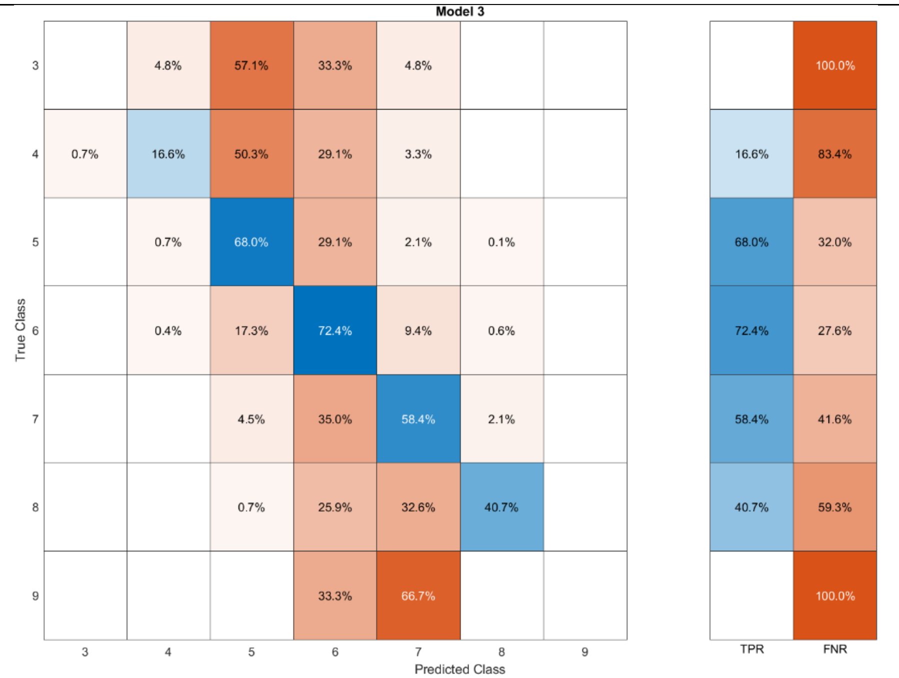
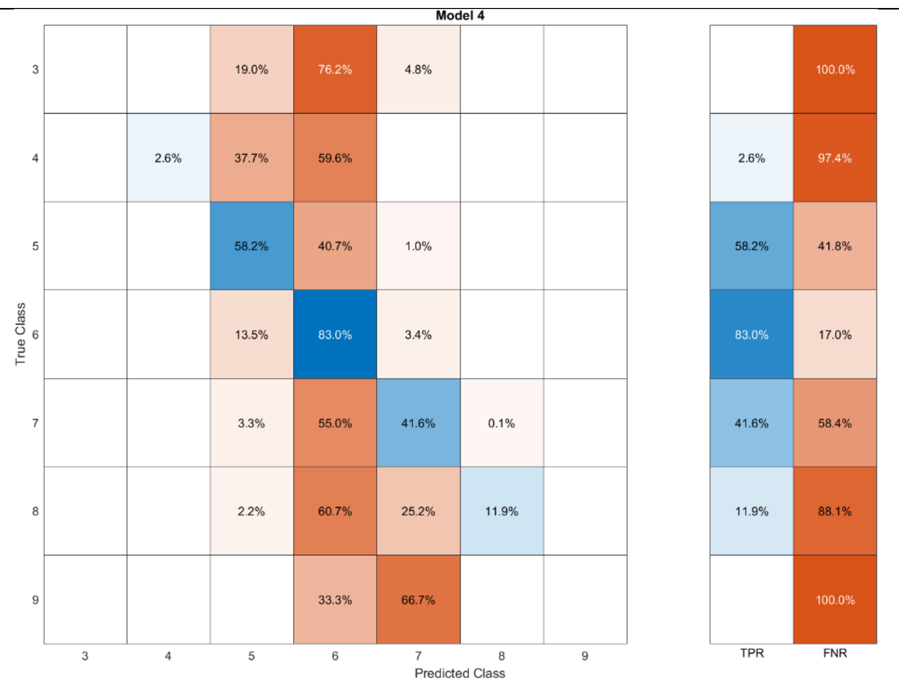
Table 4. *Models' validation vs official submission accuracy*

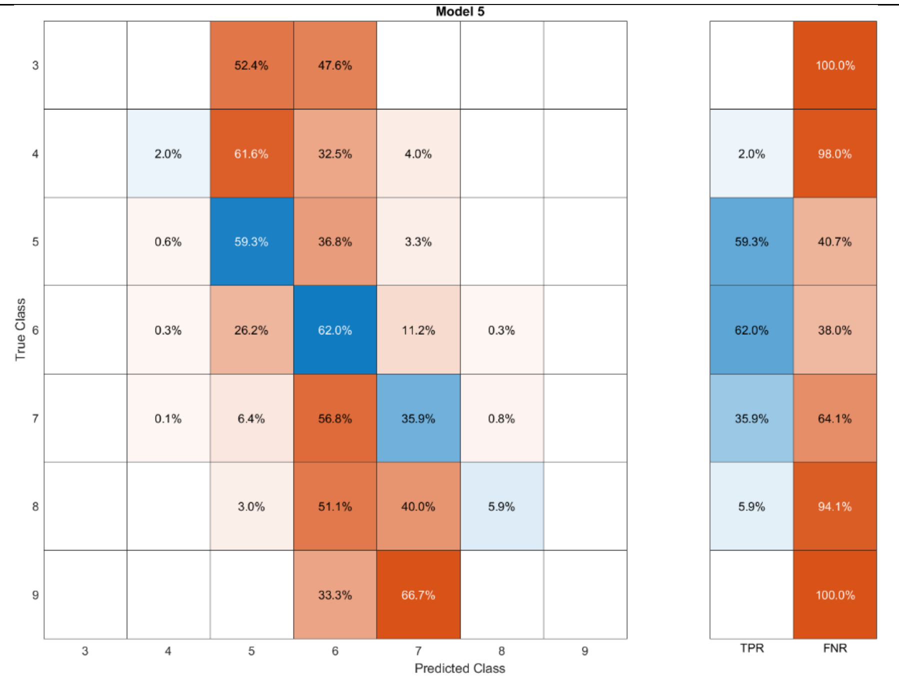
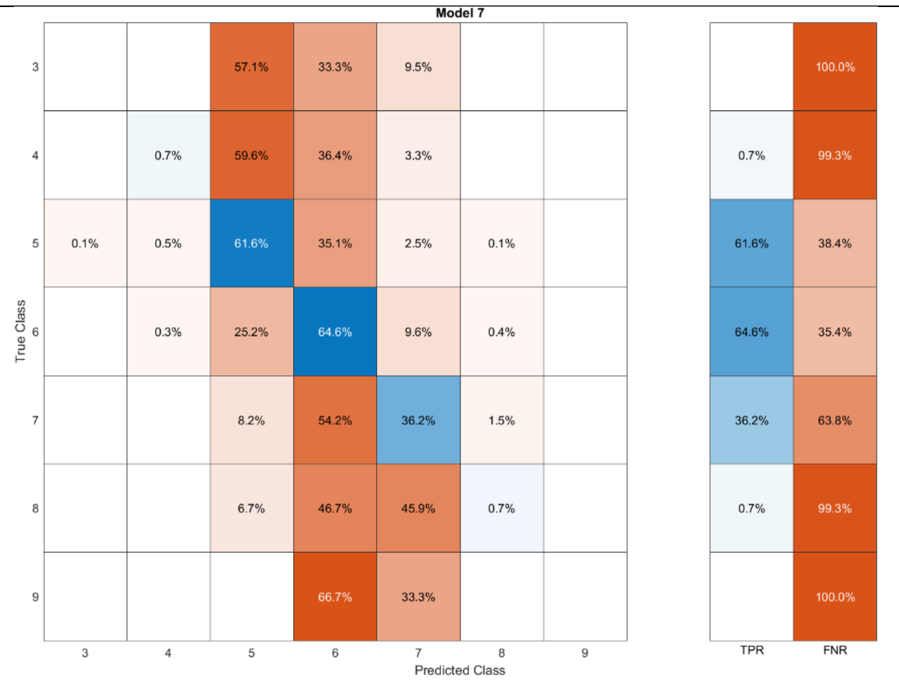
Index	Model Used	Validation Accuracy (VA)	Actual Accuracy
1	Bagged Trees (default)	0.669	0.53617
2	Bagged Trees (learners = 100)	0.677	0.55720
3	Weighted KNN (default)	0.654	0.47870
4	Fine Gaussian SVM (default)	0.628	0.48435
5	Fine Tree (default)	528	0.50846
6	Cubic KNN (default)	0.545	0.50384
7	RUS Boosted Trees (default)	0.327	0.30938
8	Boosted Trees (default)	0.532	0.54232
9	Weighted KNN (k = 68)	0.680	0.53873
10	Trilayered Neural Net (default)	0.541	0.53771
11	Medium Neural Net (default)	0.553	0.52591
12	Coarse Gaussian SVM (default)	0.524	0.53976
13	Bagged Trees (learners = 250)	0.679	0.55823
14	Coarse KNN (default)	0.533	0.53822
15	Bagged Trees (learners = 68)	0.673	0.55053 (Rank 3)
16	Bagged Trees (learners = 500)	0.683	0.56695 (Rank 1)
17	Bagged Trees (learners = 1000)	0.680	0.56541 (Rank 2)

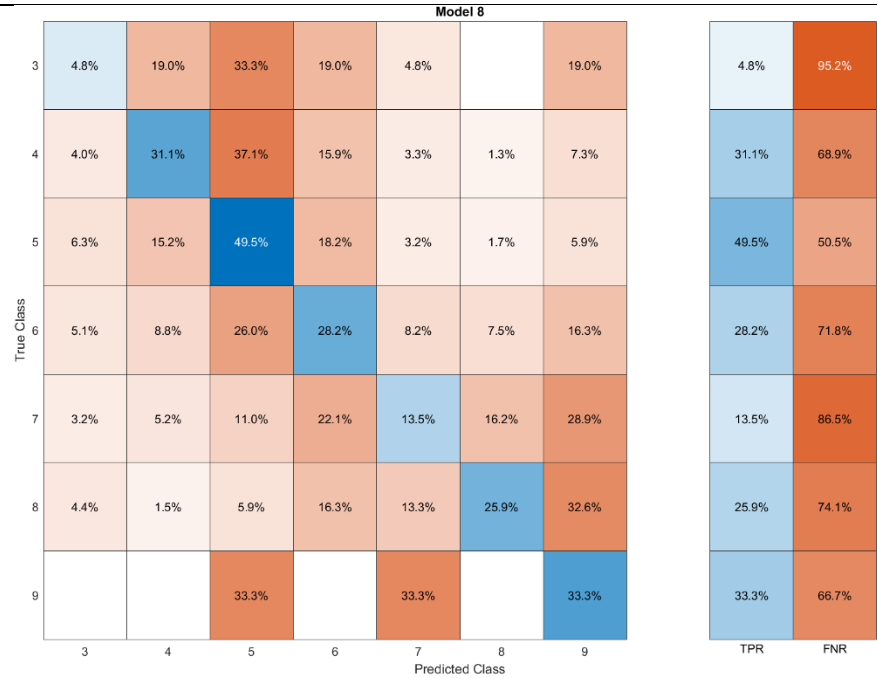
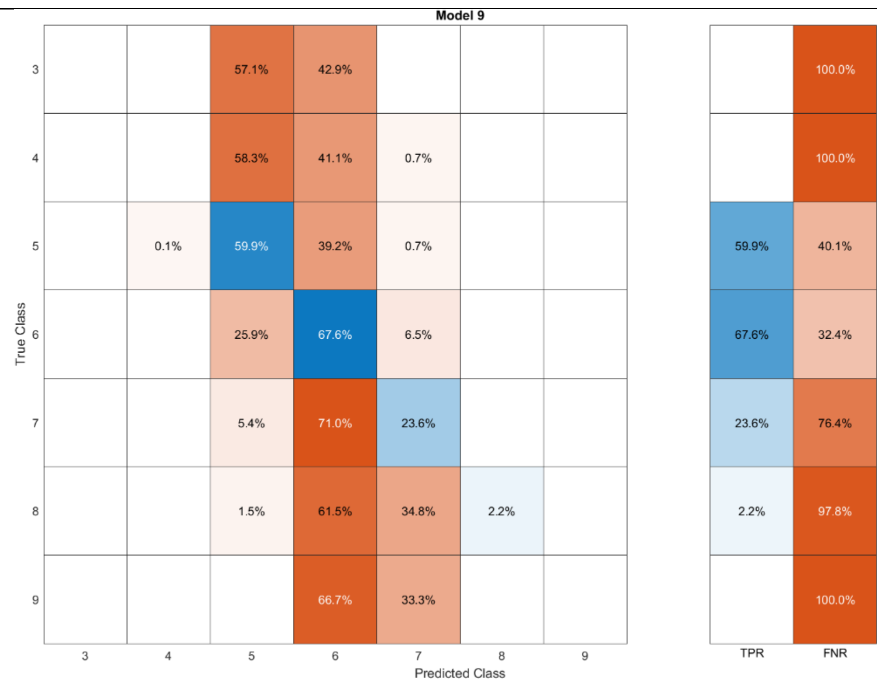
b. Confusion Matrices of models

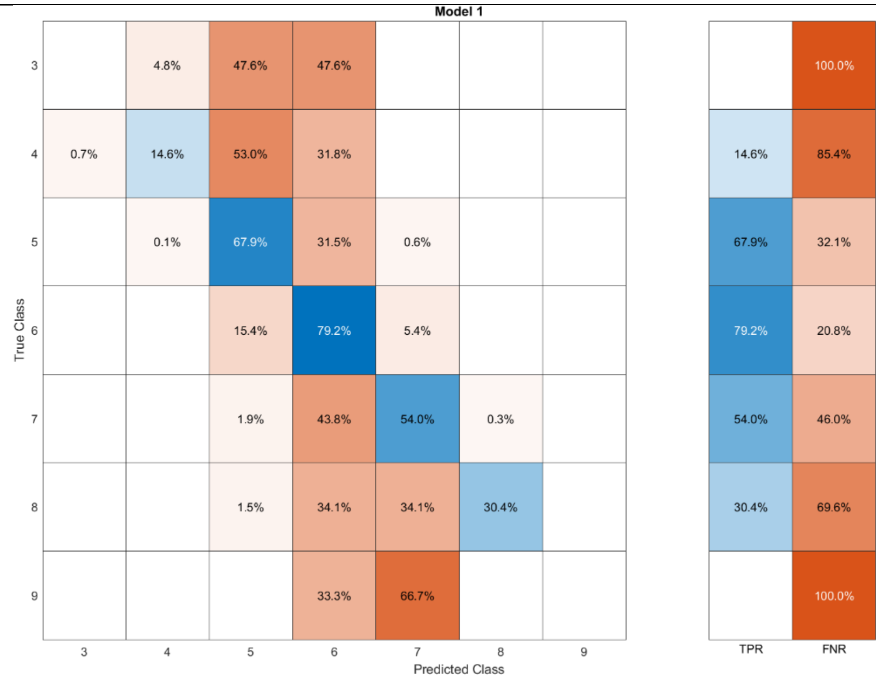
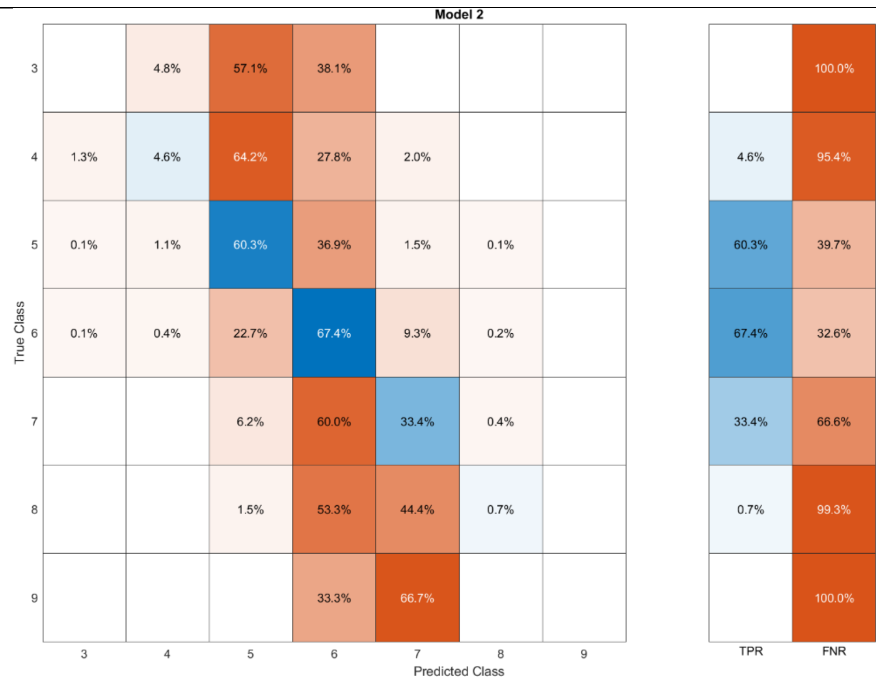
Table 5. *Trained models' validation confusion matrices*

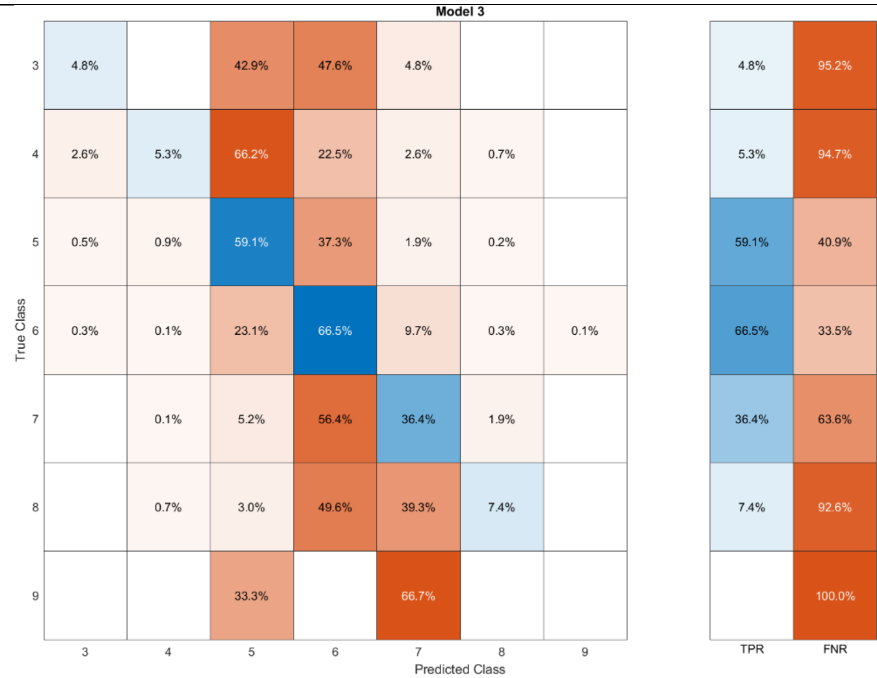
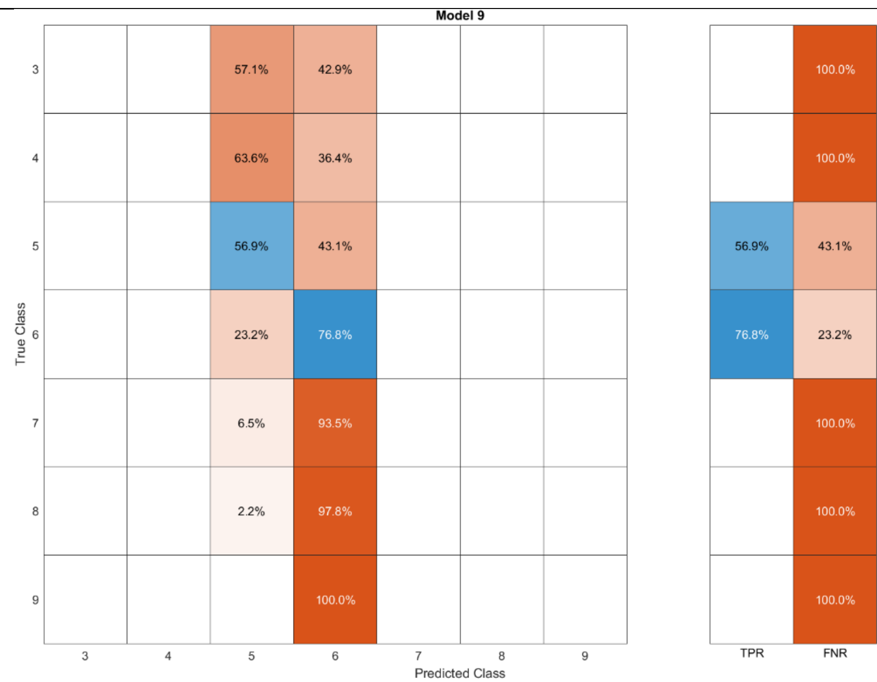
<table>
<tr><th>Index</th><th>Model Used</th><th>VA</th><th>Validation Confusion Matrices</th></tr>
<tr><td>1</td><td>Bagged Trees (default)</td><td>0.669</td><td>
</td></tr>
<tr><td>2</td><td>Bagged Trees (learners = 100)</td><td>0.677</td><td>
</td></tr>
</table>

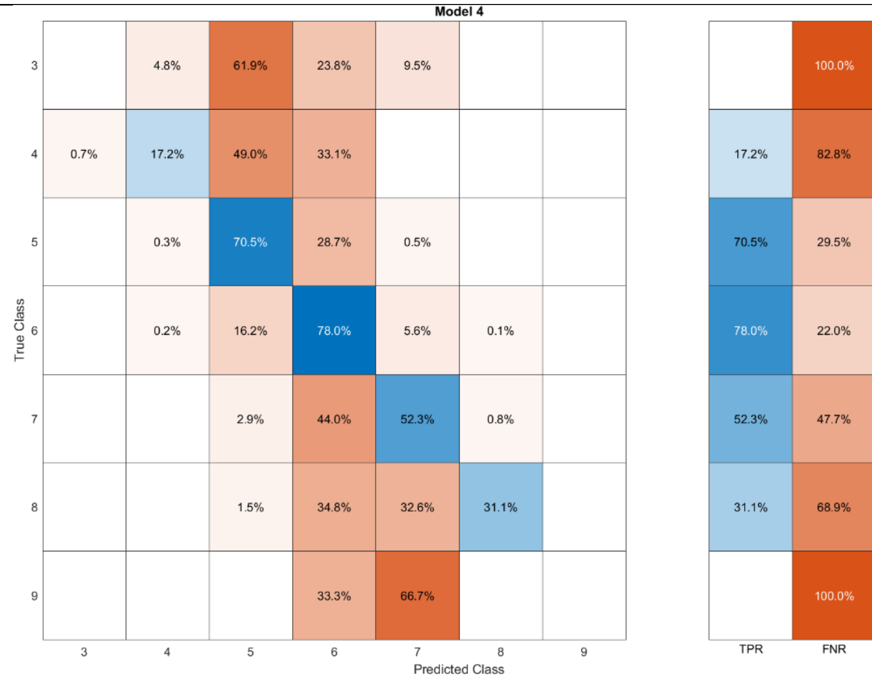
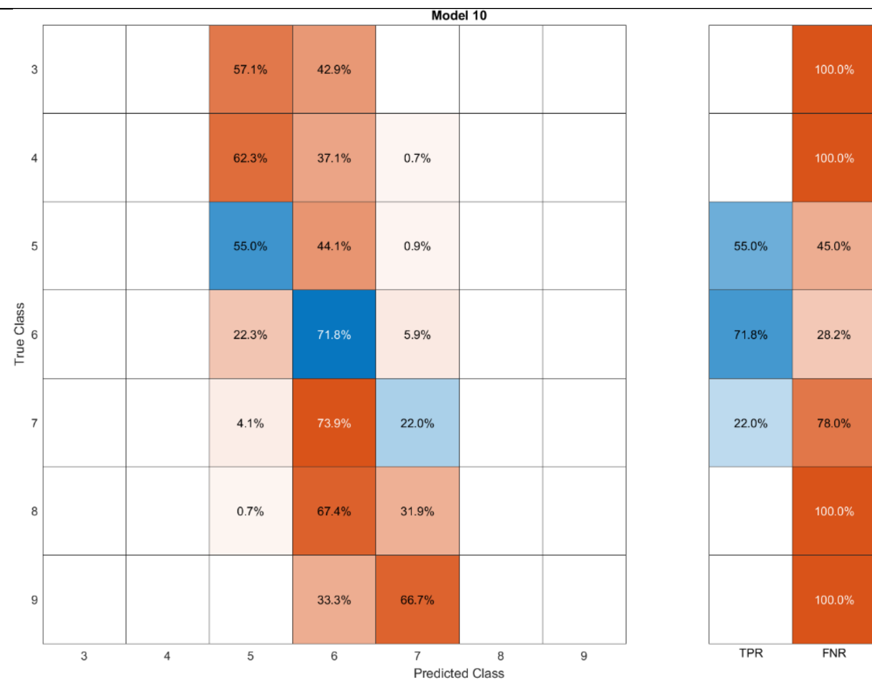
<table>
<tr><td>3</td><td>Weighted KNN (default)</td><td>0.654</td><td>
</td></tr>
<tr><td>4</td><td>Fine Gaussian SVM (default)</td><td>0.628</td><td>
</td></tr>
</table>

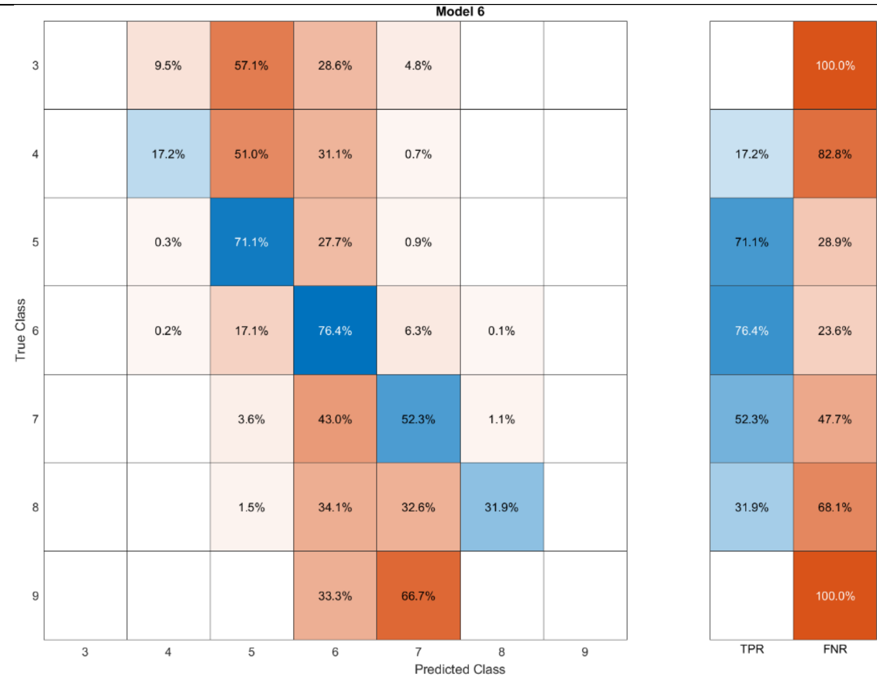
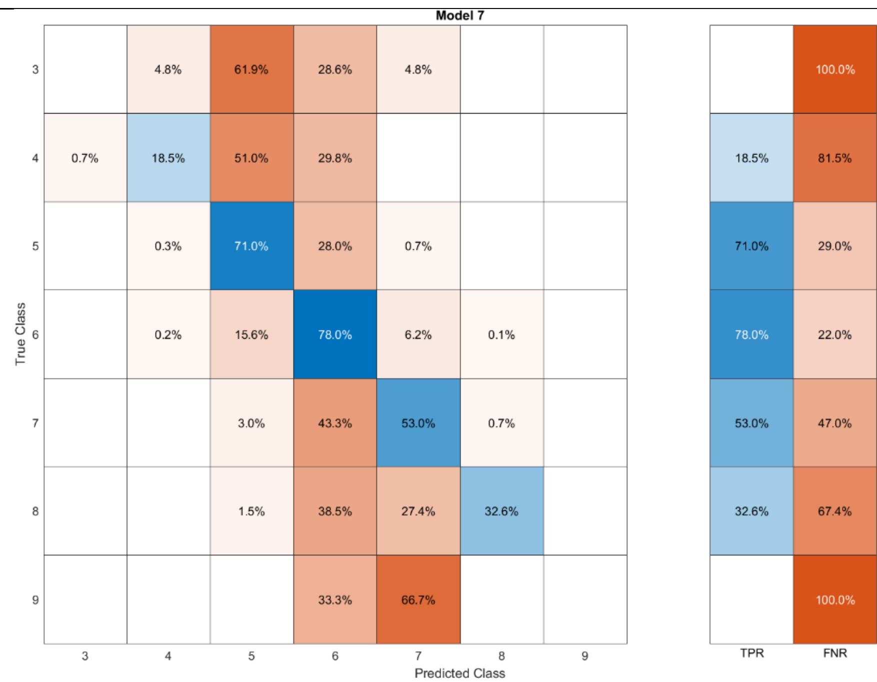
<table>
<tr><td>5</td><td>Fine Tree (default)</td><td>528</td><td>
</td></tr>
<tr><td>6</td><td>Cubic KNN (default)</td><td>0.545</td><td>
</td></tr>
</table>

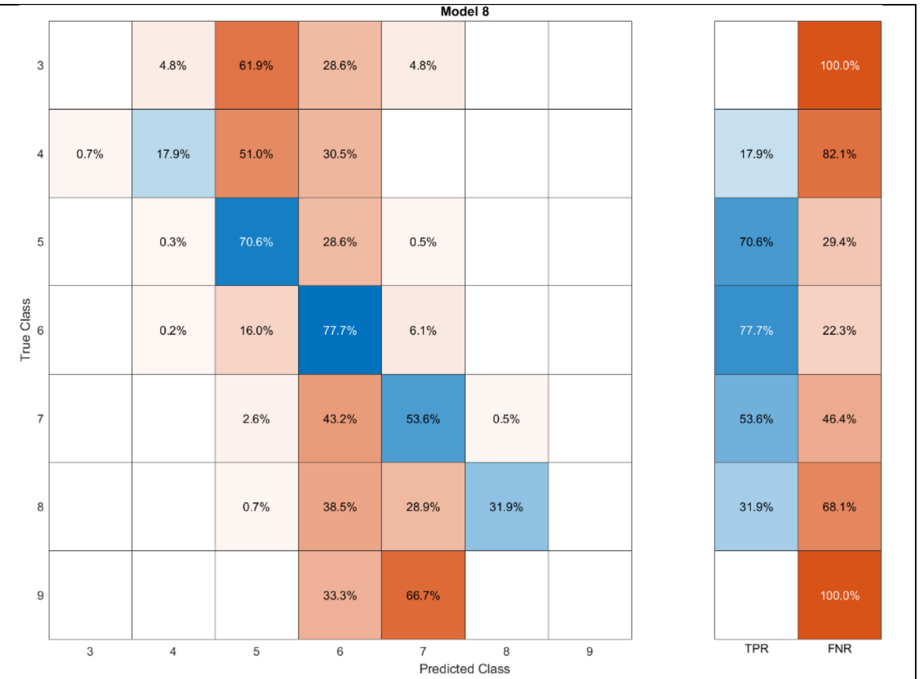
<table>
<tr><td>7</td><td>RUS Boosted Trees (default)</td><td>0.327</td><td>
</td></tr>
<tr><td>8</td><td>Boosted Trees (default)</td><td>0.532</td><td>
</td></tr>
</table>

<table>
<tr>
<td>9</td>
<td>Weighted KNN (k = 68)</td>
<td>0.680</td>
<td>
</td>
</tr>
<tr>
<td>10</td>
<td>Trilayered Neural Net (default)</td>
<td>0.541</td>
<td>
</td>
</tr>
</table>

11	Medium Neural Net (default)	0.553	
12	Coarse Gaussian SVM (default)	0.524	

13	Bagged Trees (learners = 250)	0.679	
14	Coarse KNN (default)	0.533	

15	Bagged Trees (learners = 68)	0.673	
16	Bagged Trees (learners = 500)	0.683	

17	Bagged Trees (learners = 1000)	0.680	

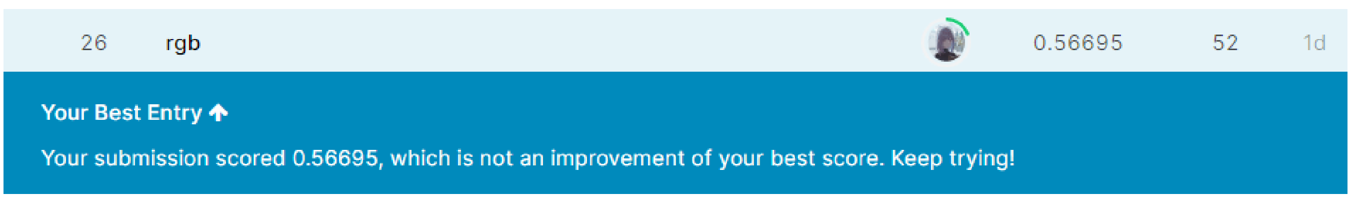


Figure 8. *Official Kaggle Submission with a 0.56695 accuracy*

V. Appendix

a. **Appendix A.** *Live Script for training separate Red and White wine types into fitnet with hidden layers of [9 2 2 5]*

```
clc; clearvars;
% Read data set for wine quality classification train and test
wine_train = readtable('train.csv');
wine_test  = readtable('test.csv');
```

```
% change wine to index
wine_train.wineType = grp2idx(wine_train.wineType)
wine_test.wineType  = grp2idx(wine_test.wineType)
```

```
tabulate(wine_train.wineType)
```

```
% Remove id
test_id              = wine_test.id;
wine_train.id        = [];
wine_test.id         = [];
```

```
tabulate(wine_train.quality)
```

```
% Get sample each
quality_3 = wine_train(wine_train.quality == 3, :);
quality_4 = wine_train(wine_train.quality == 4, :);
quality_5 = wine_train(wine_train.quality == 5, :);
quality_6 = wine_train(wine_train.quality == 6, :);
quality_7 = wine_train(wine_train.quality == 7, :);
quality_8 = wine_train(wine_train.quality == 8, :);
quality_9 = wine_train(wine_train.quality == 9, :);
```

```
% Get k samples based from each quality clusters
k = 1000;
quality_3s = datasample(quality_3, k);
quality_4s = datasample(quality_4, k);
quality_5s = wine_train(wine_train.quality == 5, :)
quality_6s = wine_train(wine_train.quality == 6, :)
quality_7s = wine_train(wine_train.quality == 7, :)
% quality_5s = datasample(quality_5, k);
% quality_6s = datasample(quality_6, k);
% quality_7s = datasample(quality_7, k);
quality_8s = datasample(quality_8, k);
quality_9s = datasample(quality_9, k);
```

```
% Combine all new data
new_wine_train = [quality_3s;quality_4s; quality_5s;
quality_6s;quality_7s;quality_8s;quality_9s];
% Tabulate updated quality
tabulate(new_wine_train.quality)
```

```
% Target quality
red_wine_target =   new_wine_train(new_wine_train.wineType == 1,
"quality")
red_wine_target = table2array(red_wine_target)
white_wine_target = new_wine_train(new_wine_train.wineType == 2,
"quality")
white_wine_target = table2array(white_wine_target)
```

```
% Separate train and test
```

```
red_train = new_wine_train(new_wine_train.wineType == 1, :)
white_train = new_wine_train(new_wine_train.wineType == 2, :)
red_test = wine_test(wine_test.wineType == 1, :)
white_test = wine_test(wine_test.wineType == 2, :)
```

```
% Normalize train and test
red_wine_train    = normalize(removevars(red_train, {'quality',
'wineType'}), 'range');
white_wine_train    = normalize(removevars(white_train, {'quality',
'wineType'}), 'range');
red_wine_test    = normalize(removevars(red_test, {'wineType'}), 'range');
white_wine_test    = normalize(removevars(white_test, {'wineType'}),
'range');
```

```
red_wine_train
[idx,scores]    = fscchi2(red_wine_train, red_wine_target)
bar(scores(idx))
xlabel('Predictor rank')
ylabel('Predictor importance score')
```

```
white_wine_train
[idx2,scores2]    = fscchi2(white_wine_train, white_wine_target)
bar(scores2(idx2))
xlabel('Predictor rank')
ylabel('Predictor importance score')
```

Test

```
% id = test_id;
% quality = BGTREE88.predictFcn(wine_test_fin);
% results = table(id, quality);
% writetable(results , 'matlabSubmission12.csv');
```

```
% RED Feature, Target, Testing param
red_ft = idx(1:7)
red_feature = red_wine_train{:, red_ft} % idx(1:9)
red_target = double(red_wine_target)
red_testing = red_wine_test{:, red_ft} % idx(1:9)


red_x = red_feature(:,:)';
red_y = red_target';
red_z = red_testing(:,:)';
```

```
tabulate(red_y)
```

```
% WHITE Feature, Target, Testing param
white_ft = idx2(1:7)
white_feature = white_wine_train{:,white_ft} % idx(1:9)
white_target = double(white_wine_target)
white_testing = white_wine_test{:, white_ft} % idx(1:9)


white_x = white_feature(:,:)';
white_y = white_target';
white_z = white_testing(:,:)';
tabulate(white_y)
```

```
redNet = fitnet([9 2 2 5], 'trainlm');
redNet.trainParam.max_fail = 10;
redTrainNet = train(redNet, red_x, red_y);
```

```
whiteNet = fitnet([9 2 2 5], 'trainlm');
whiteNet.trainParam.max_fail = 10;
whiteTrainNet = train(whiteNet, white_x, white_y);
```

```
red_results = round(redTrainNet(red_z))
tabulate(red_y)
tabulate(red_results)
```

```
white_results = round(whiteTrainNet(white_z))
tabulate(white_y)
tabulate(white_results)
```

Save the results to .csv

```
id = test_id;
quality = [red_results, white_results]'
tabulate(quality)
exported = table(id, quality);
writetable(exported , 'results.csv');
```

b. **Appendix B**. *Final Live Script for Wine Quality Classification*

LBYEC3B Project 1 - BAYETA | TAN | TUPAL

Wine Quality Classification

```
% Clearvars
clc; clearvars;
```

```
% Read data set for wine quality classification train and test
wine_train = readtable('train.csv');
wine_test  = readtable('test.csv');
```

Statistical Analysis

```
% Change wineType from str to idx
% wine_train.wineType = grp2idx(wine_train.wineType);
% wine_test.wineType  = grp2idx(wine_test.wineType);
```

```
% Remove ID
test_id             = wine_test.id;
qlty                = wine_train.quality;
wine_train.id       = [];
wine_test.id        = [];
wine_train.wineType = [];
wine_test.wineType  = [];
```

```
% Displays the wine features
colnames = wine_train.Properties.VariableNames
figure('Name','Wine Features', 'NumberTitle', 'off')
for idx = 1: length(colnames)
    col_name = string(colnames(:,idx));
        subplot(4,3,idx)
        histogram(wine_train.(col_name))
        title(col_name)
        % disp(idx)
end
```

```
summary(wine_train)
```

```
% Generate correlation matrix
figure()
corrplot(wine_train)
```

```
% Display colnames
colnames
```

Preprocessing

```
% Feature ranking by chi-squared test
% Univariate feature ranking for classification using chi-square tests
% fscchi2 examines whether each predictor variable is independent of
% a response variable by using individual chi-square tests.
figure()
[idx2,scores2]    = fscchi2(removevars(wine_train, {'quality'}), qlty)
bar(scores2(idx2))
xlabel('Predictor rank')
ylabel('Predictor importance score')
```

```
% Tabulate idx2 vs scores2
top_col = idx2';
top_col_scores = scores2';
idx2vsscores2 = table(top_col, top_col_scores);
idx2vsscores2
```

```
% Display colnames
colnames
```

```
% Drop quality
wine_train.quality = [];
% Functions for outliers (did not work)
% new_data = filloutliers(wine_train, 'spline','percentile', [10 90]);
```

```
% Normalize
norm_train = normalize(wine_train);
norm_test  = normalize(wine_test);
```

```
% Select ranked features
mask = idx2(1:8)
wine_feature = norm_train{:, mask};
wine_target = double(qlty);
wine_testing = norm_test{:, mask};
```

Models and Results

Models are trained under the Classification App of MATLAB. It can be replicated by checking the parameters stated in each section below.

```
% % Bagged Trees (Random-Forest), Default Params
% id = test_id;
% quality = FBGTREE_DEF_669.predictFcn(wine_testing);
% disp('Quality Prediction for Final1')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final1_baggedtrees_def_669.csv');
```

```
% % Bagged Trees (Random-Forest), 100 learners
% id = test_id;
% quality = FBGTREE_100L_677.predictFcn(wine_testing);
% disp('Quality Prediction for Final2')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final2_baggedtrees_100L_677.csv');
```

```
% % Weighted KNN, Default Params
% id = test_id;
% quality = FWKNN_DEF_654.predictFcn(wine_testing);
% disp('Quality Prediction for Final3')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final3_weightedKNN_def_654.csv');
```

```
% % Fine Gaussian SVM, Default Params
% id = test_id;
% quality = FFGSVM_DEF_628.predictFcn(wine_testing);
% disp('Quality Prediction for Final4')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final4_finegaussianSVM_def_628.csv');
```

```
% % Fine Tree, default parameters
% id = test_id;
% quality = FFTREE_DEF_528.predictFcn(wine_testing);
% disp('Quality Prediction for Final5')
% tabulate(quality)
% exported = table(id, quality);
```

```
% writetable(exported , 'final5_finetree_def_528.csv');
```

```
% % Cubic KNN, default
% id = test_id;
% quality = FCKNN_DEF_545.predictFcn(wine_testing);
% disp('Quality Prediction for Final6')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final6_cubicKNN_def_545.csv'); % 7 in window
```

```
% % RUS Boosted Trees, default parameters
% id = test_id;
% quality = FRUSBST_DEF_327.predictFcn(wine_testing);
% disp('Quality Prediction for Final7')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final7_RUSBoostedTrees_def_327.csv');
```

```
% % Boosted Trees, default
% id = test_id;
% quality = FBSTREE_DEF_532.predictFcn(wine_testing);
% disp('Quality Prediction for Final8')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final8_BoostedTrees_def_532.csv');
```

```
% % Weighted KNN, k = 68
% id = test_id;
% quality = FWKNN_68K_680.predictFcn(wine_testing);
% disp('Quality Prediction for Final9')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final9_WKNN_68k_680.csv');
```

```
% % Trilayered NN, default
% id = test_id;
% quality = WTRINN_DEF_541.predictFcn(wine_testing);
% disp('Quality Prediction for Final10')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final10_TRINN_541.csv');
```

```
% % Medium NN, default
% id = test_id;
```

```
% quality = FMEDNN_DEF_553.predictFcn(wine_testing);
% disp('Quality Prediction for Final11')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final11_MEDNN_553.csv');
```

```
% % Coarse Gaussian SVM
% id = test_id;
% quality = FCGSVM_DEF_524.predictFcn(wine_testing);
% disp('Quality Prediction for Final12')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final12_CoarseGaussSVM_524.csv');
```

```
% % Bagged Trees (Random Forest), 250 learners
% id = test_id;
% quality = FBGTREE_250L_679.predictFcn(wine_testing);
% disp('Quality Prediction for Final13')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final13_baggedtrees_250L_679.csv');
```

```
% % Coarse KNN, default
% id = test_id;
% quality = FCKNN_DEF_533.predictFcn(wine_testing);
% disp('Quality Prediction for Final14')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final14_coarseKNN_def_533.csv');
%
```

```
% % Bagged Trees (Random Forest), 68 learners
% id = test_id;
% quality = FBGTREE_68L_673.predictFcn(wine_testing);
% disp('Quality Prediction for Final15')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final15_baggedtrees_68L_673.csv');
```

```
% % Bagged Trees (Random Forest), 500 learners
% disp('nag run ako')
% id = test_id;
% quality = FBGTREE_500L_683.predictFcn(wine_testing);
% disp('Quality Prediction for Final16')
% tabulate(quality)
```

```
% exported = table(id, quality);
% writetable(exported , 'final16_baggedtrees_500L_683.csv');
```

```
% % Bagged Trees (Random Forest), 1000 learners
% id = test_id;
% quality = FBGTREE_1000L_680.predictFcn(wine_testing);
% disp('Quality Prediction for Final17')
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'final17_baggedtrees_1000L_680.csv');
```

```
% id = test_id;
% quality = BAGTREE5K66.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'lastsubmission3.csv');
```

```
% id = test_id;
% quality = BAGTREE5k668.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'lastsubmission4.csv');
```

```
% id = test_id;
% quality = WKK5k65.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'lastsubmission5.csv');
```

```
% id = test_id;
% quality = DEFBGTREE68.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'lastsubmission6.csv');
```

```
% id = test_id;
% quality = DEFWKNN669.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% writetable(exported , 'lastsubmission7.csv');
```

```
% id = test_id;
% quality = BAGTREESIDX1_67.predictFcn(wine_testing);
```

```
% tabulate(quality)
% exported = table(id, quality);
% %idx1 (11 param)
% writetable(exported , 'lastsubmission8.csv');
```

```
% id = test_id;
% quality = bg66idx8.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param)
% writetable(exported , 'lastsubmission9.csv');
```

```
% id = test_id;
% quality = bg671.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission10.csv');
```

```
% id = test_id;
% quality = bgg655id2x17.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission11.csv');
```

```
% id = test_id;
% quality = BAGTREEUPDATED677IDX28.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission12.csv');
```

```
% id = test_id;
% quality = bg673idx19.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission13.csv');
```

```
% id = test_id;
% quality = bg658idx18.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
```

```
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission14.csv');
```

```
% id = test_id;
% quality = bg67idx18.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission15.csv');
```

```
% id = test_id;
% quality = trainedModelbg100677idx28.predictFcn(wine_testing);
% tabulate(quality)
% exported = table(id, quality);
% %idx2 (9 param 1-9)
% writetable(exported , 'lastsubmission16.csv');
```