Project 2

The epicenter of the earthquake and the surrounding major cities in Haiti.

UTC time 2010-01-12 21:53

Gladies. Huilin chang (hc5hq)

Background

- A real historical data-mining problem, locating displaced persons living in makeshift shelters following the destruction caused by earthquake in Haiti in 2010.
- People whose homes had been destroyed by the earthquake were creating temporary shelters using blue tarps.
- The goal was to find the best algorithm that could search images and locate displaced persons in time for the locations to be communicated back to rescue workers.
- Total dataset is two: one is a smaller dataset with 5 classes, the other is a bigger dataset with 2 classes.

Conclusions

- **10X CV cross-validation**
  - Recommend to adopt **KNN** model which has the highest sensitivity rate (96.54%). The reason and the purpose of this study is to predict the "blue tarp" correctly. This means the higher the true positive rate, the greater the accuracy. Since "blue tarp" proportion is only 3%, sensitivity(recall) is adopted as index for model selection.
  - The ranking of various models from sensitivity rates are KNN (96.54%) > Random Forest (94.36%) > SVM (93.67%) > Logistic Regression (91.34%) > QDA (86.88%) > LDA (80.45%)
  - Clearly, KNN model shows the best in accuracy, recall, F measure and AUC (K=13)
  - The true negative rate (> 99%) are high for all models due to blue tarp being only 3.2% in total.

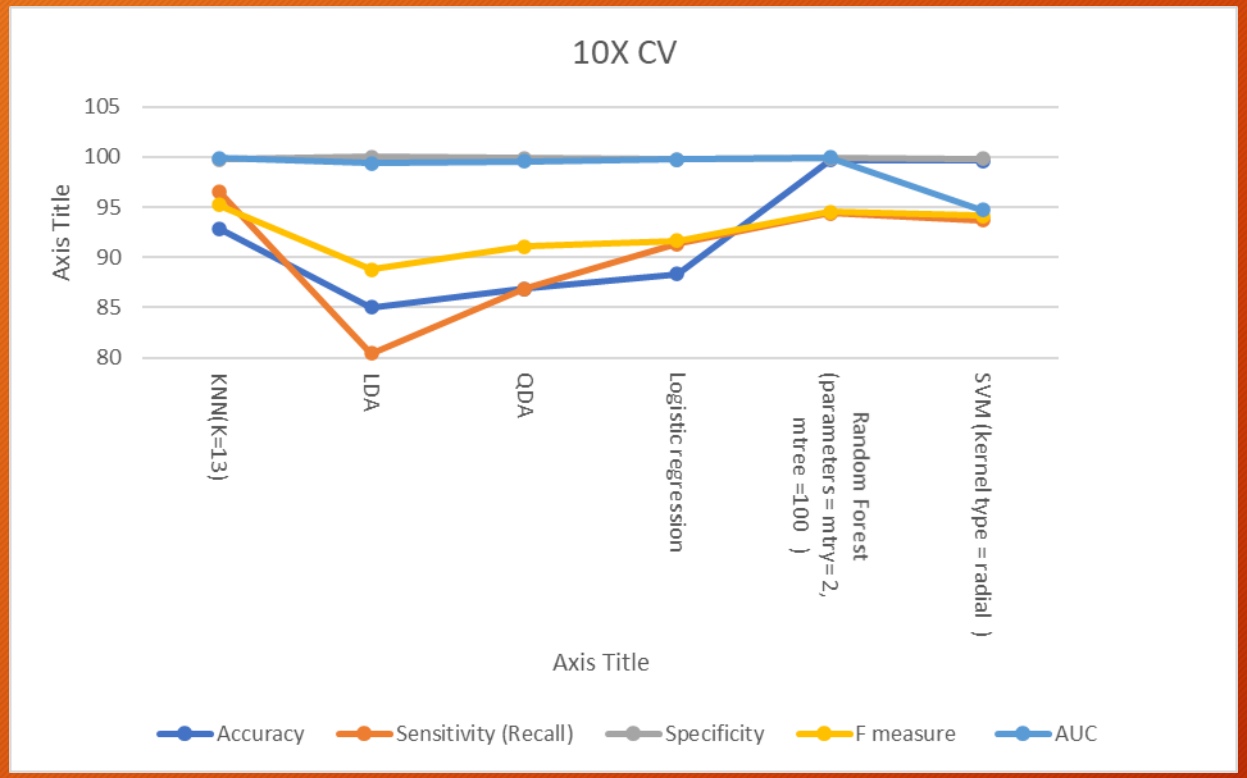
Conclusions

- **Hold-Out data**
  - Recommend to adopt **SVM** model which has the highest sensitivity rate (99.26%).
  - The ranking of various models from sensitivity rates are SVM (99.26%) > Random Forest (98.89%) > KNN (98.80%) > Logistic Regression (97.84%) > LDA (57.08%) > QDA (54.51%)
  - The true negative rate (> 99%) are high for all models due to blue tarp being only 1% in total.

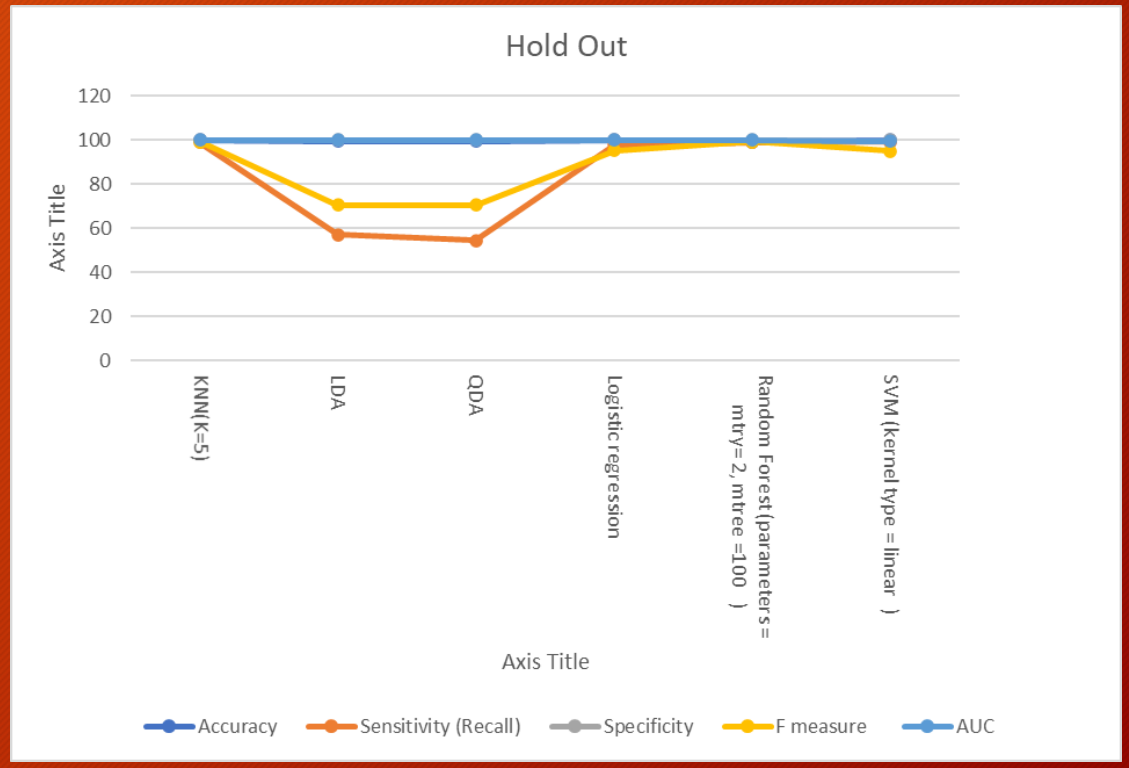
Conclusions

- Data structure for Model selection matters. The 10x CV validation demonstrates KNN is the best model. In contrast, the hold-out demonstrates SVM is the best. The possible explanation of the discrepancy is population of blue tarp.
  - KNN offers the flexibility to adjust the nearest neighbor, the blue tarp is 3 % in 10X CV validation set, the optimum K value can be distinguished
  - However, the hold out data, the blue tarp is only 1%, this means the majority of the blue tarp's neighbors are non-blue tarp, the small K in KNN is favored but causes increases variation.
  - SVM stands out in Hold out data, because kernel selection offers the benefits of unbalanced dataset.

Summary Charts



10X CV



Hold out

10x Cross-Validation Performance Matrix

	Accuracy	Sensitivity (Recall)	Specificity	F measure	AUC
KNN(K=13)	92.89	96.54	99.72	95.23	99.85
LDA	85.05	80.45	99.98	88.80	99.35
QDA	86.88	86.88	99.87	91.05	99.55
Logistic regression	88.36	91.34	99.74	91.68	99.75
Random Forest (parameters = mtry= 2, mtree =100)	99.69	94.36	99.86	94.52	99.92
SVM (kernel type = radial)	99.63	93.67	99.83	94.16	94.70

Hold Out Performance matrix

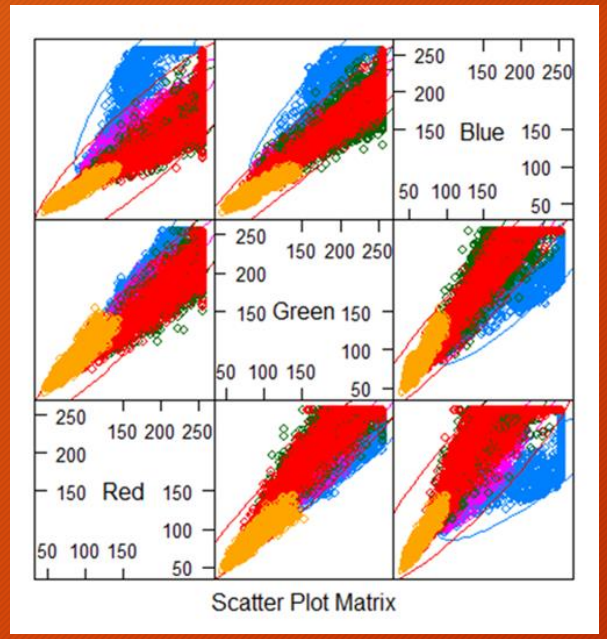
	Accuracy	Sensitivity (Recall)	Specificity	F measure	AUC
KNN(K=5)	99.99	98.80	100	99.20	99.80
LDA	99.33	57.08	99.93	70.58	99.84
QDA	99.28	54.51	100	70.55	99.95
Logistic regression	99.92	97.84	99.93	95.10	99.98
Random Forest (parameters = mtry= 2, mtree =100)	99.20	98.89	100	99.20	99.88
SVM (kernel type = linear)	99.92	99.26	99.92	94.96	99.77

Model Selections

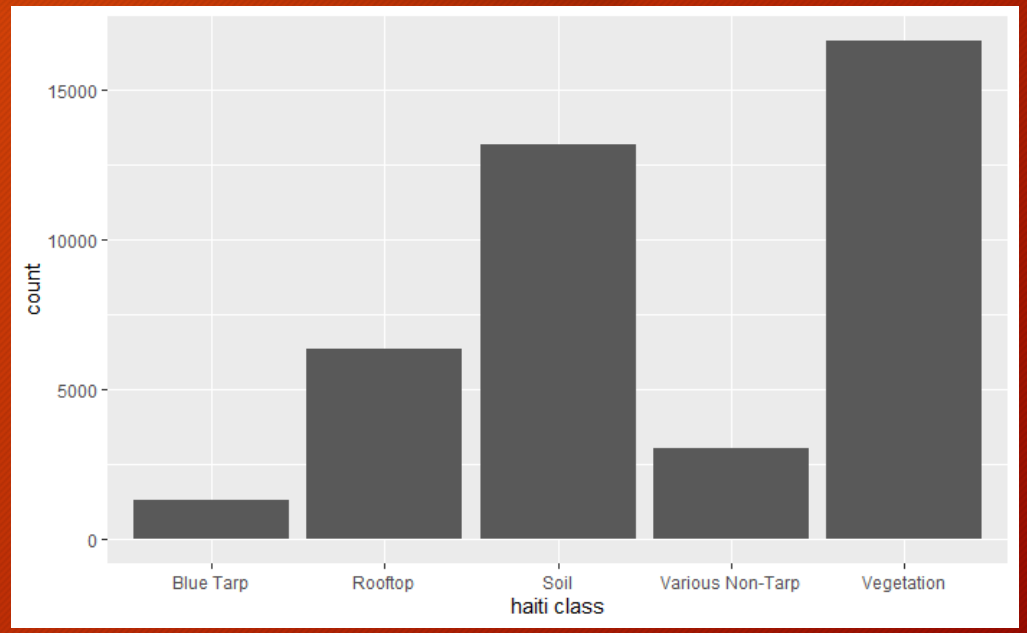
10X CROSS VALIDATION

Model Selections

- **Nature of data**
  - dim(data) = 63241 * 4
  - Class = five
- **Model Considerations**
  - KNN
  - LDA
  - QDA
  - Logistic Regression
  - SVM
  - Random Forest
- **Two approaches**
  - Broaden the class because the concern is blue tarp specific
  - Keep five classes



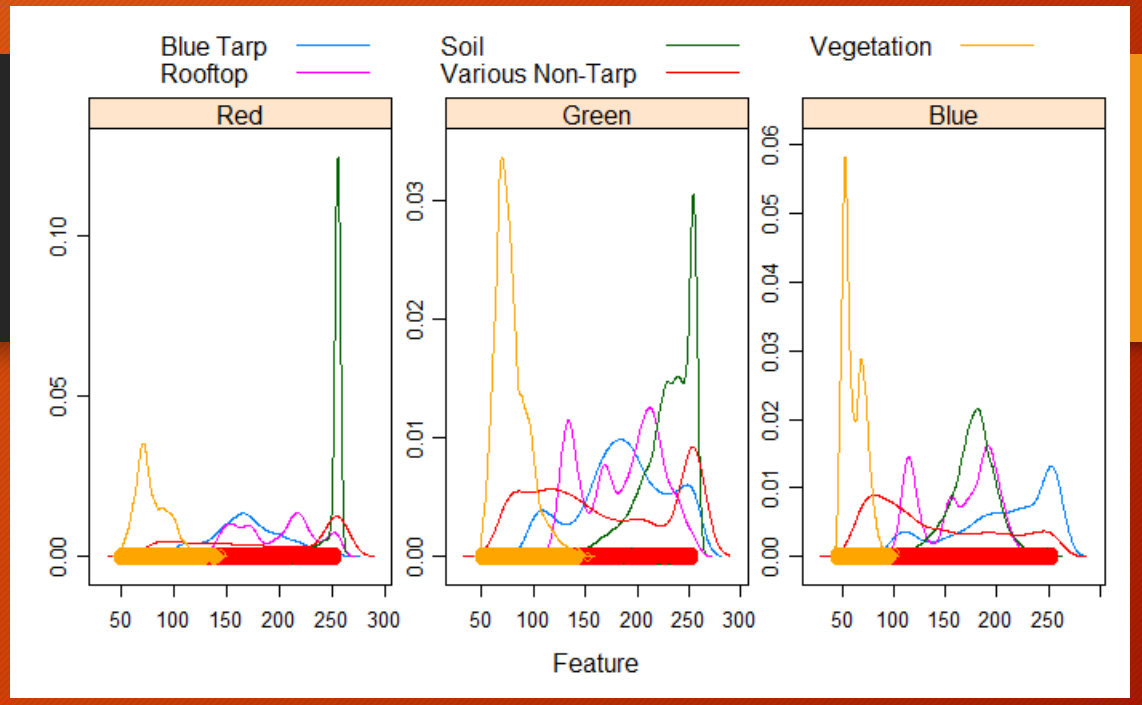
Pair-wise feature scatter plots



The distribution of 5 classes

5 classes on features understanding

- Blue Tarp: 3.20%
- Rooftop: 15.66%
- Soil: 21.52%
- Various Non-Tarp: 7.50%
- Vegetation: 41.12%



	freq	percentage
Blue Tarp	1618	3.197944
Rooftop	7923	15.65965
Soil	16453	32.519024
Various Non-Tarp	3796	7.502718
Vegetation	20805	41.120664

The details of 5 classes

K-nearest neighbors (KNN)

- Accuracy: 0.9285 (K=13)
- Sensitivity: 0.9654
- Specificity: 0.9971
- The greatest accuracy occurs at K=13

Confusion Matrix and Statistics

	Reference					
Prediction	Blue Tarp	Rooftop	Soil	Various	Non-Tarp	Vegetation
Blue Tarp	390	26	8		1	0
Rooftop	10	1873	80		92	0
Soil	0	73	3899		283	0
Various Non-Tarp	4	8	125		441	62
Vegetation	0	0	1		131	5139

Overall Statistics

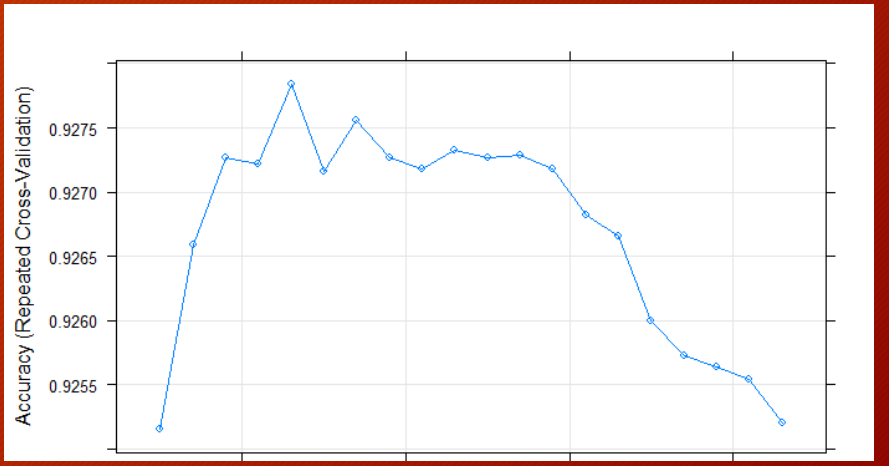
Accuracy	0.9285
95% CI	(0.9239, 0.9329)
No Information Rate	0.4113
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.8962
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: Blue Tarp	Class: Rooftop	Class: Soil	Class: Various Non-Tarp	Class: Vegetation
Sensitivity	0.96535	0.9460	0.9480	0.46519	0.9881
Specificity	0.99714	0.9829	0.9583	0.98299	0.9823
Pos Pred Value	0.91765	0.9114	0.9163	0.68906	0.9750
Neg Pred Value	0.99885	0.9899	0.9745	0.95777	0.9916
Prevalence	0.03195	0.1566	0.3252	0.07496	0.4113
Detection Rate	0.03084	0.1481	0.3083	0.03487	0.4064
Detection Prevalence	0.03361	0.1625	0.3365	0.05061	0.4168
Balanced Accuracy	0.98124	0.9644	0.9531	0.72409	0.9852

Confusion matrix

Model summary table



K value vs. Model accuracy

Linear Discriminant Analysis (LDA)

- Accuracy: 0.8505
- Sensitivity: 0.8044
- Specificity: 0.9997

Confusion Matrix and Statistics

Prediction	Reference: Blue Tarp	Rooftop	Soil	Various Non-Tarp	Vegetation
Blue Tarp	325	1	2	0	0
Rooftop	35	1274	197	208	0
Soil	0	230	3823	323	0
Various Non-Tarp	0	434	67	134	1
Vegetation	44	41	24	283	5200

Confusion matrix

Overall Statistics

Accuracy	0.8505
95% CI	(0.8442, 0.8567)
No Information Rate	0.4113
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.7801
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: Blue Tarp	Class: Rooftop	Class: Soil	Class: Various Non-Tarp	Class: Vegetation
Sensitivity	0.80446	0.6434	0.9295	0.14135	0.9998
Specificity	0.99975	0.9587	0.9352	0.95709	0.9473
Pos Pred Value	0.99085	0.7433	0.8736	0.21069	0.9299
Neg Pred Value	0.99359	0.9354	0.9649	0.93222	0.9999
Prevalence	0.03195	0.1566	0.3252	0.07496	0.4113
Detection Rate	0.02570	0.1007	0.3023	0.01060	0.4112
Detection Prevalence	0.02594	0.1355	0.3460	0.05029	0.4422
Balanced Accuracy	0.90211	0.8011	0.9323	0.54922	0.9736

Model summary table

Quadratic Discriminant Analysis (QDA)

Confusion Matrix and Statistics

Prediction	Reference: Blue Tarp	Rooftop	Soil	Various	Non-Tarp	Vegetation
Blue Tarp	351	13	3	0		0
Rooftop	7	1750	129	182		3
Soil	0	166	3843	290		0
Various Non-Tarp	46	51	138	260		27
Vegetation	0	0	0	216		5171

Overall Statistics

Accuracy	0.8995
95% CI	(0.8941, 0.9047)
No Information Rate	0.4113
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.8532
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: Blue Tarp	Class: Rooftop	Class: Soil	Class: Various Non-Tarp	Class: Vegetation
Sensitivity	0.86881	0.8838	0.9344	0.27426	0.9942
Specificity	0.99869	0.9699	0.9466	0.97760	0.9710
Pos Pred Value	0.95640	0.8450	0.8939	0.49808	0.9599
Neg Pred Value	0.99568	0.9783	0.9677	0.94325	0.9959
Prevalence	0.03195	0.1566	0.3252	0.07496	0.4113
Detection Rate	0.02776	0.1384	0.3039	0.02056	0.4089
Detection Prevalence	0.02902	0.1638	0.3399	0.04128	0.4260
Balanced Accuracy	0.93375	0.9269	0.9405	0.62593	0.9826

Confusion matrix

Model summary table

- Accuracy: 0.8995
- Sensitivity: 0.8688
- Specificity: 0.9987

LOGISTIC REGRESSION

- Accuracy: 0.8836
- Sensitivity: 0.9134
- Specificity: 0.9974

Confusion Matrix and Statistics

Prediction	Reference: Blue Tarp	Rooftop	Soil	Various Non-Tarp	Vegetation
Blue Tarp	369	26	5	1	0
Rooftop	35	1642	208	297	18
Soil	0	290	3864	341	0
Various Non-Tarp	0	22	35	127	11
Vegetation	0	0	1	182	5172

Overall Statistics

Accuracy	0.8836
95% CI	(0.8779, 0.8891)
No Information Rate	0.4113
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.829
Mcnemar's Test P-Value	NA

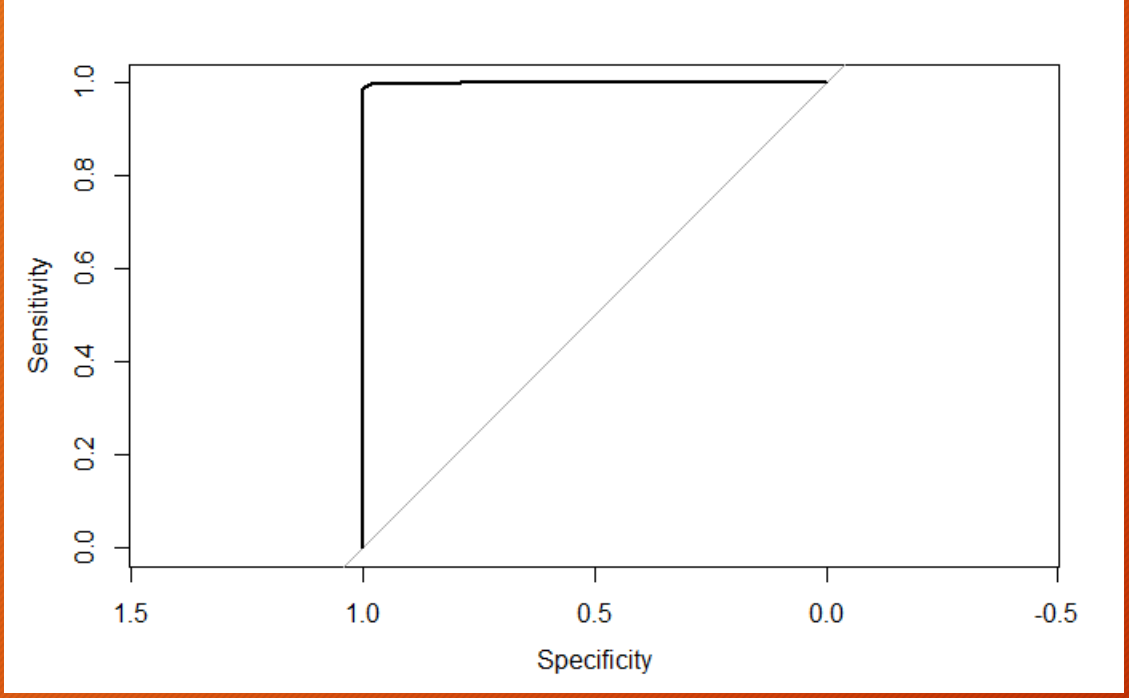
Confusion matrix

Statistics by Class:

	Class: Blue Tarp	Class: Rooftop	Class: Soil	Class: Various Non-Tarp	Class: Vegetation
Sensitivity	0.91337	0.8293	0.9395	0.13397	0.9944
Specificity	0.99739	0.9477	0.9261	0.99419	0.9754
Pos Pred Value	0.92020	0.7464	0.8596	0.65128	0.9658
Neg Pred Value	0.99714	0.9676	0.9695	0.93406	0.9960
Prevalence	0.03195	0.1566	0.3252	0.07496	0.4113
Detection Rate	0.02918	0.1298	0.3056	0.01004	0.4090
Detection Prevalence	0.03171	0.1740	0.3554	0.01542	0.4235
Balanced Accuracy	0.95538	0.8885	0.9328	0.56408	0.9849

Model summary table

Random Forest (RF)



ROC curve

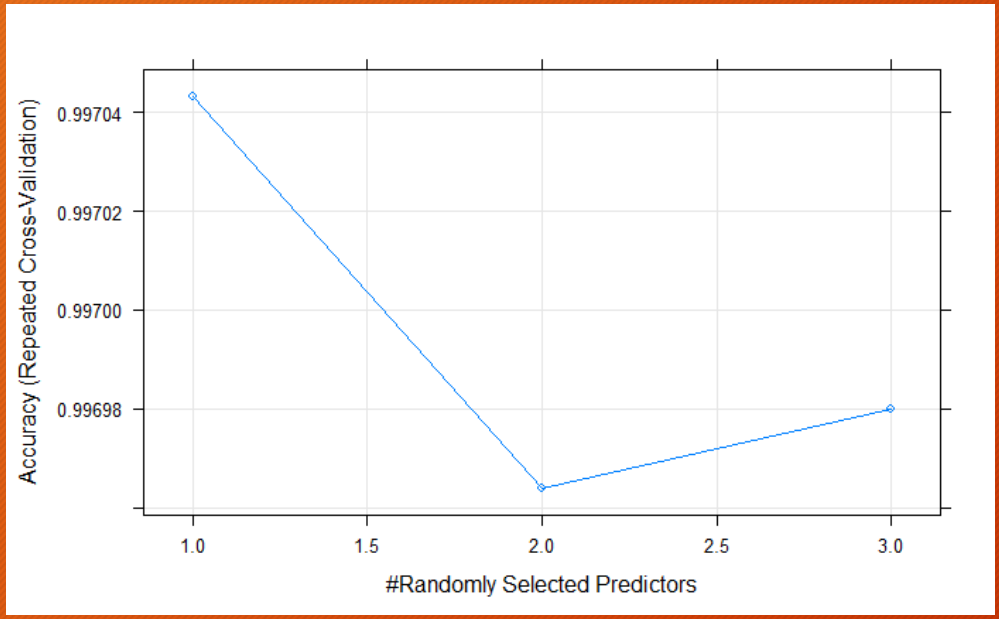
Confusion matrix

```
Confusion Matrix and Statistics

          Reference
Prediction  blue not_blue
  blue      1917       85
  not_blue   105    61134

               Accuracy : 0.997
                 95% CI : (0.9965, 0.9974)
    No Information Rate : 0.968
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9512

 Mcnemar's Test P-Value : 0.1681

            Sensitivity : 0.94807
            Specificity : 0.99861
         Pos Pred Value : 0.95754
         Neg Pred Value : 0.99829
             Prevalence : 0.03197
         Detection Rate : 0.03031
   Detection Prevalence : 0.03166
      Balanced Accuracy : 0.97334

       'Positive' Class : blue
```

Model summary table

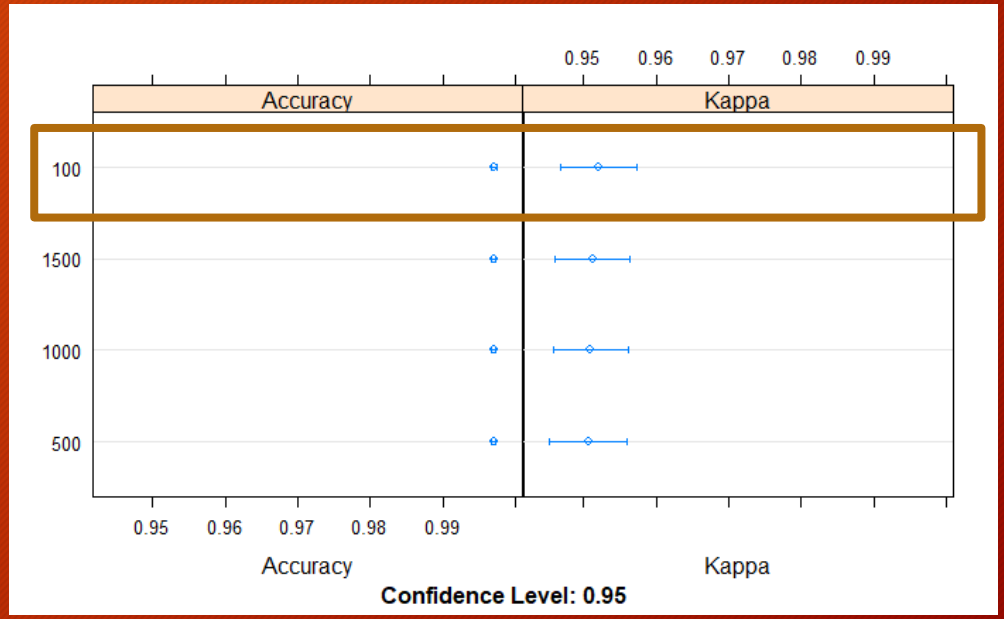
Random Forest

- Selected predictors @ 2 shows the best model accuracy
- Tree number @ 100 shows the best accuracy/Kappa



Selected predictors vs. model Accuracy

Tree numbers

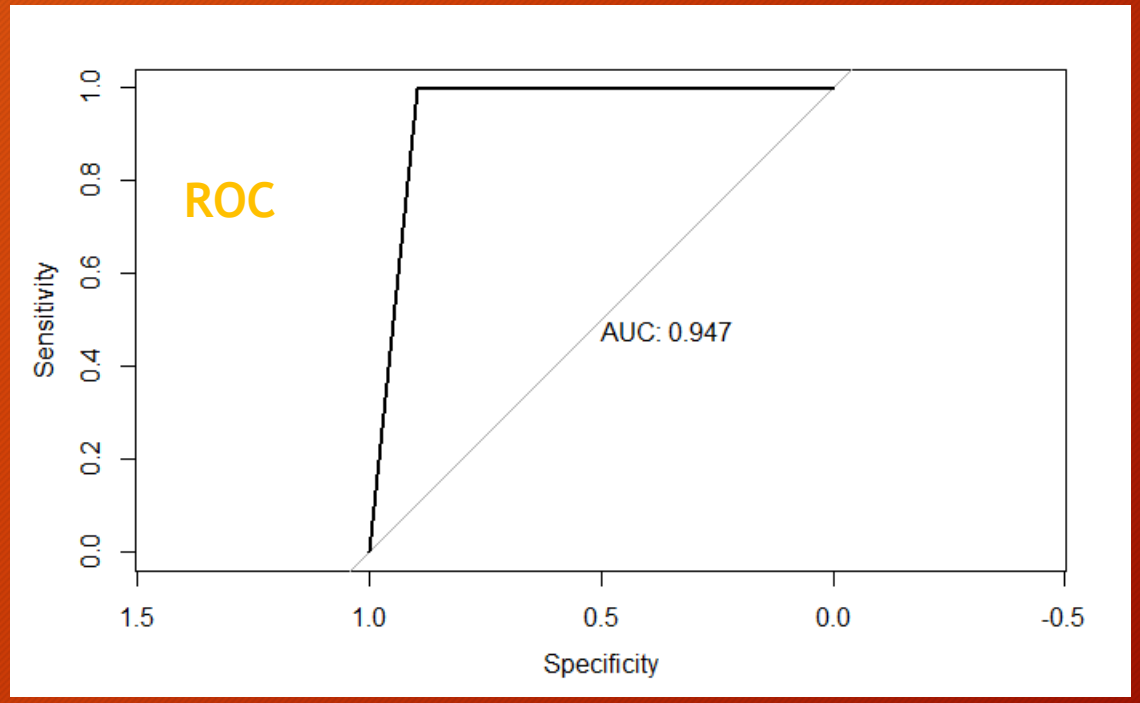


Tree numbers vs. Model accuracy

Support-vector machine (SVM)

```
Confusion Matrix and Statistics

          Reference
Prediction  blue not_blue
  blue      1894      107
  not_blue   128    61112

               Accuracy : 0.9963
                 95% CI : (0.9958, 0.9967)
    No Information Rate : 0.968
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9397

 Mcnemar's Test P-Value : 0.192

            Sensitivity : 0.93670
            Specificity : 0.99825
         Pos Pred Value : 0.94653
         Neg Pred Value : 0.99791
             Prevalence : 0.03197
         Detection Rate : 0.02995
   Detection Prevalence : 0.03164
      Balanced Accuracy : 0.96747

       'Positive' Class : blue
```

Confusion matrix

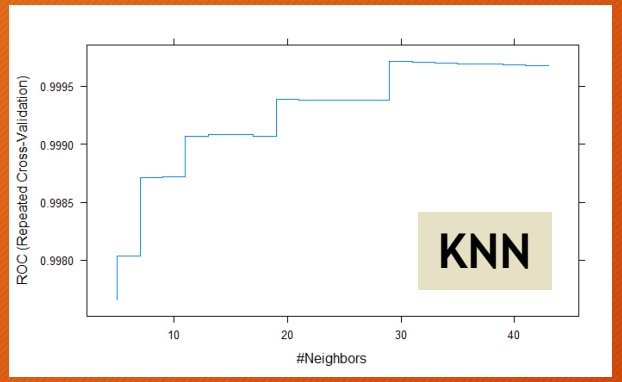


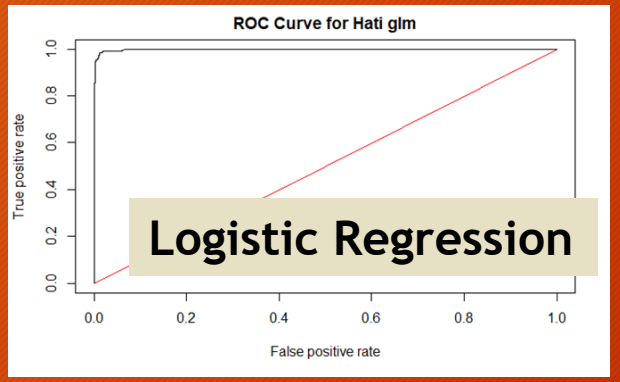
Model summary table

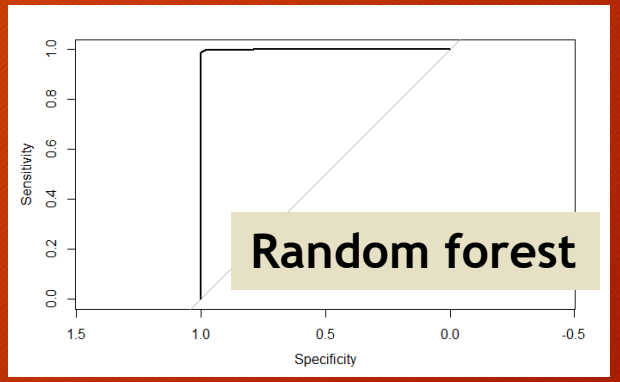
ROC

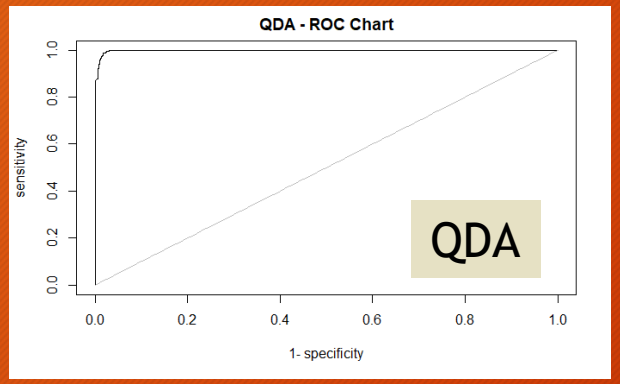
ROC Curves of six models

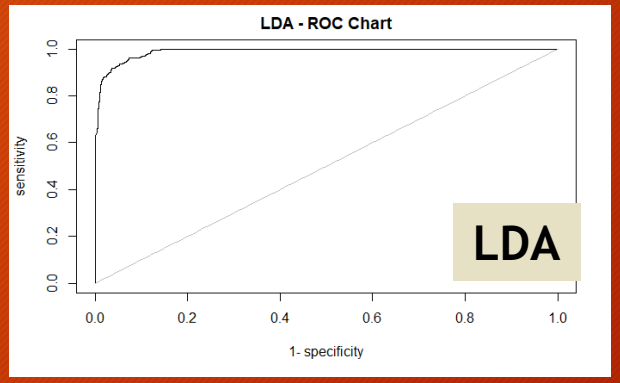
- AUC: 0.992 ~ 0.998 obtained from four models

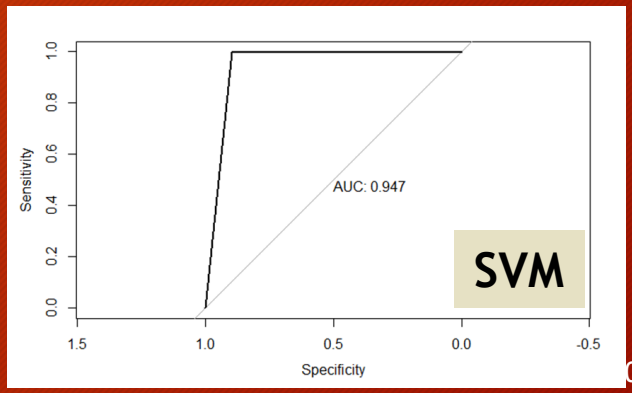












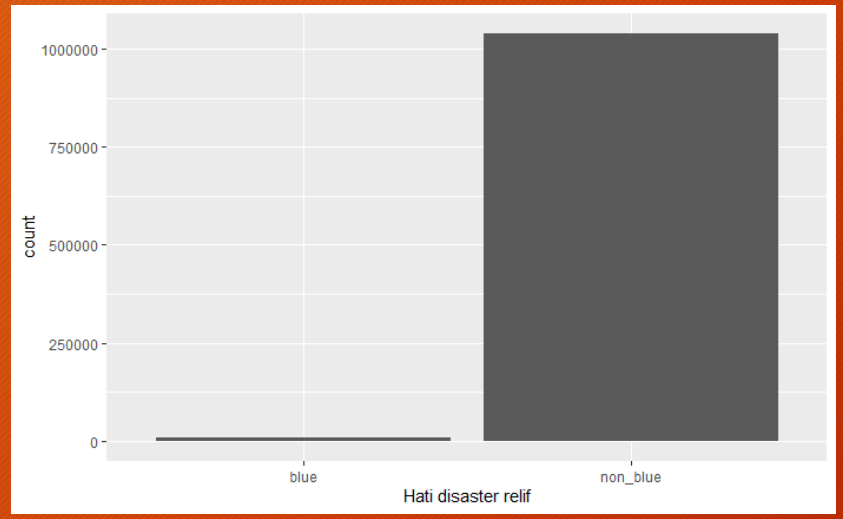
10x Cross-Validation Performance Matrix

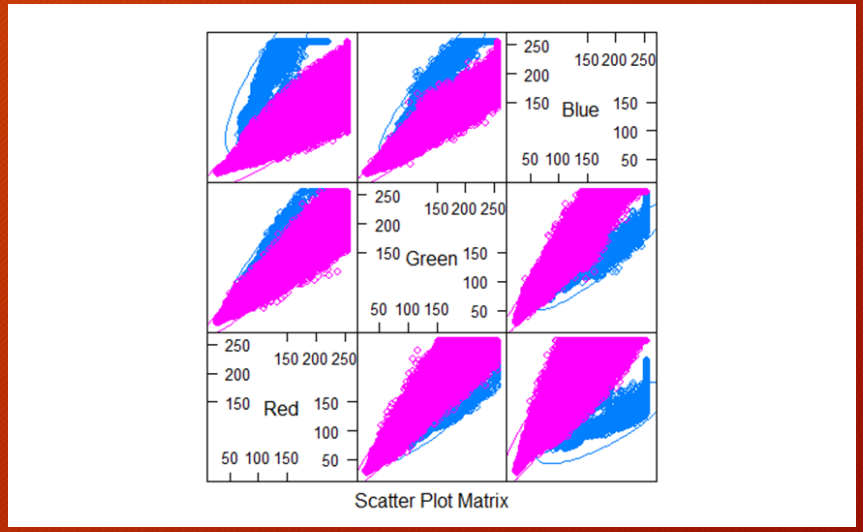
	Accuracy	Sensitivity (Recall)	Specificity	F measure	AUC
KNN(K=13)	92.89	96.54	99.72	95.23	99.85
LDA	85.05	80.45	99.98	88.80	99.35
QDA	86.88	86.88	99.87	91.05	99.55
Logistic regression	88.36	91.34	99.74	91.68	99.75
Random Forest (parameters = mtry= 2, mtree =100)	99.69	94.36	99.86	94.52	99.92
SVM (kernel type = radial)	99.63	93.67	99.83	94.16	94.70

HOLD-OUT

Hold out data

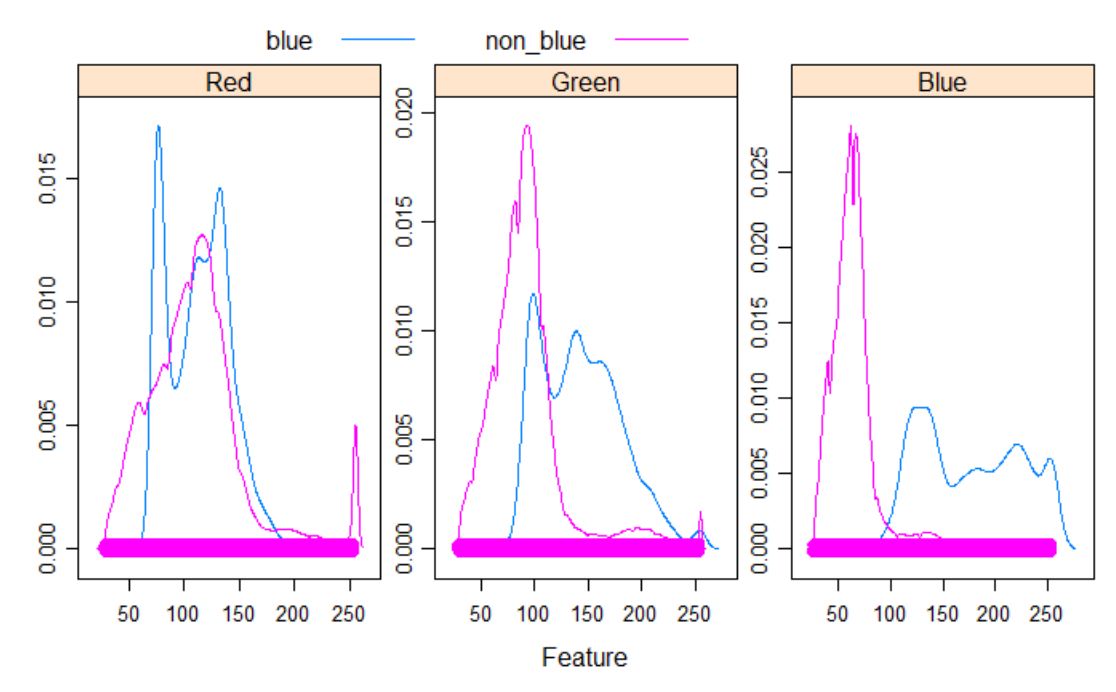
- **Nature of data**
  - Class = Two
- **Model Considerations**
  - KNN
  - LDA
  - QDA
  - Logistic Regression
  - SVM
  - Random Forest
- **Used *Python and R* to proceed hold out data**

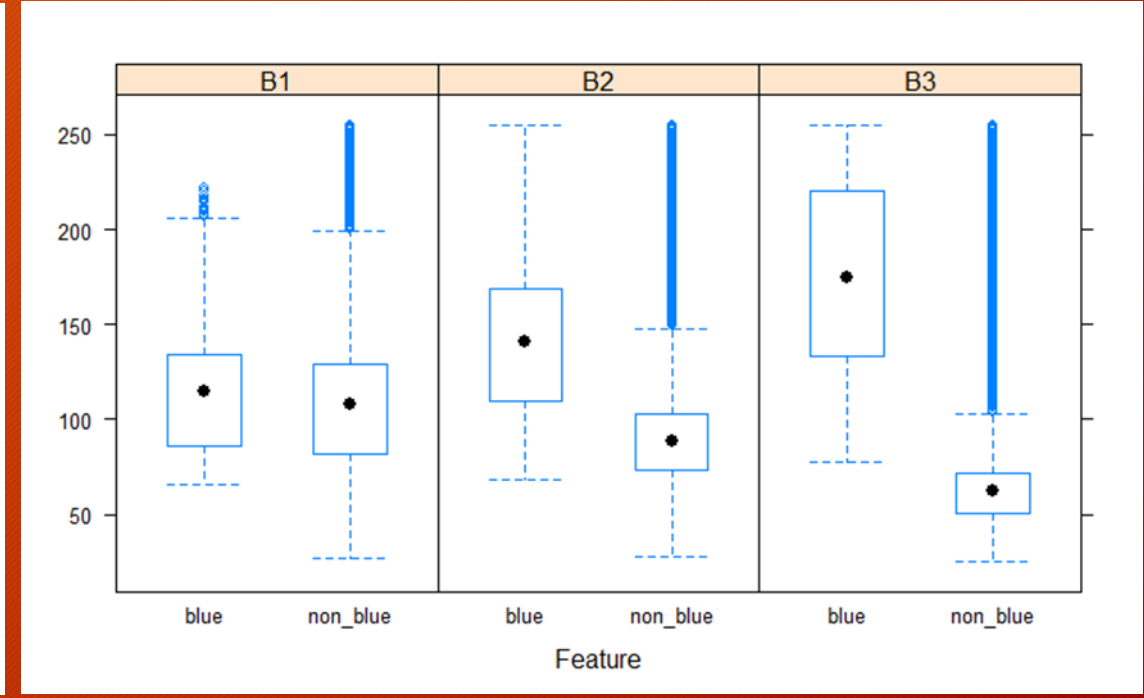




Two classes on features understanding

Feature Engineering & Distribution Understanding





Three Features (Red, Green, Blue) distribution

Hold Out Performance matrix

	Accuracy	Sensitivity (Recall)	Specificity	F measure	AUC
KNN(K=5)	99.99	98.80	100	99.20	99.80
LDA	99.33	57.08	99.93	70.58	99.84
QDA	99.28	54.51	100	70.55	99.95
Logistic regression	99.92	97.84	99.93	95.10	99.98
Random Forest (parameters = mtry= 2, mtree =100)	99.20	98.89	100	99.20	99.88
SVM (kernel type = linear)	99.92	99.26	99.92	94.96	99.77

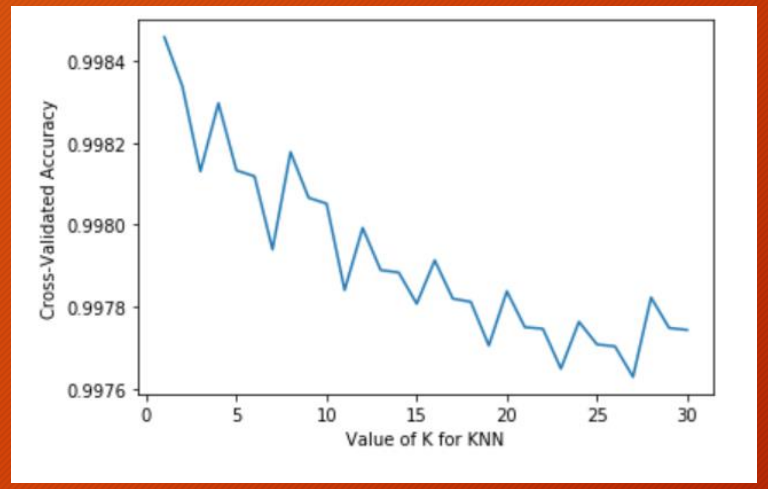
K-nearest neighbors (KNN)

```
In [19]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
         knn.fit(X_train, y_train)

Out[19]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                              metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                              weights='uniform')
```

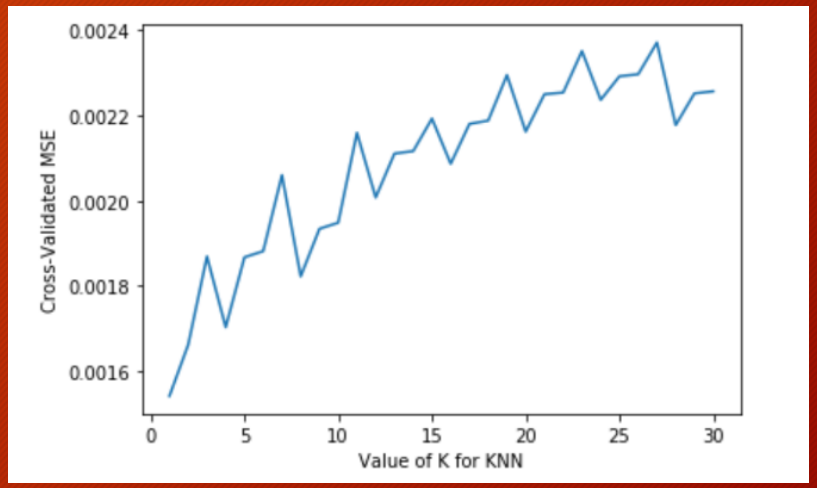


```
In [21]: y_pred = knn.predict(X_test)

In [23]: confusion_matrix(y_test, y_pred)

Out[23]: array([[  2225,      9],
                [    27, 259883]], dtype=int64)
```

Confusion Matrix (K=5)



K value vs. Model Accuracy

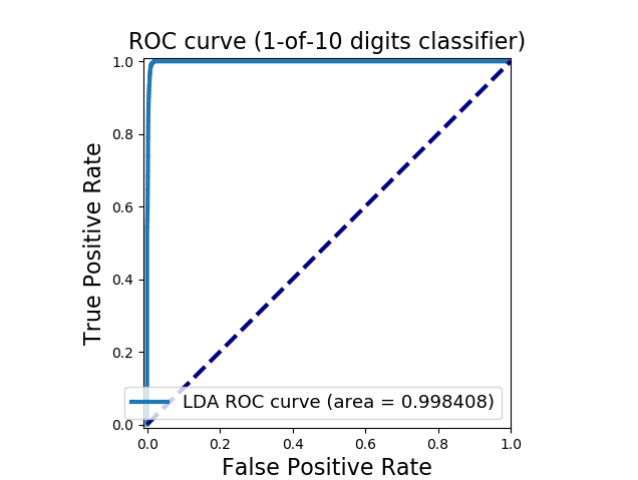


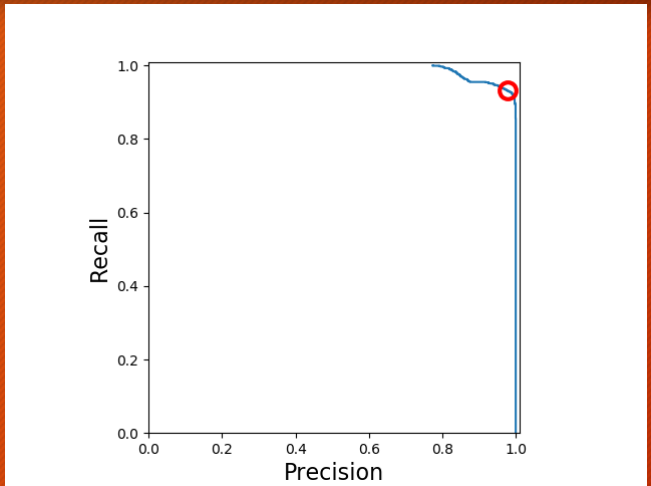
K value vs. MSE

K-nearest neighbors (KNN) model performance

Measure	Value	Derivations
Sensitivity	0.9880	TPR = TP / (TP + FN)
Specificity	1.0000	SPC = TN / (FP + TN)
Precision	0.9960	PPV = TP / (TP + FP)
Negative Predictive Value	0.9999	NPV = TN / (TN + FN)
False Positive Rate	0.0000	FPR = FP / (FP + TN)
False Discovery Rate	0.0040	FDR = FP / (FP + TP)
False Negative Rate	0.0120	FNR = FN / (FN + TP)
Accuracy	0.9999	ACC = (TP + TN) / (P + N)
F1 Score	0.9920	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.9919	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Linear Discriminant Analysis (LDA)





- ROC curve
- Recall vs. Precision
- Confusion Matrix

```
In [15]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
         from sklearn.metrics import confusion_matrix

         lda = LinearDiscriminantAnalysis().fit(X_train, y_train)
         lda_predicted = lda.predict(X_test)
         confusion = confusion_matrix(y_test, lda_predicted)

         print('Linear Discriminant Analysis classifier (default settings)\n', confusion)
```

```
Linear Discriminant Analysis classifier (default settings)
 [[309938   1901]
 [   206   2528]]
```

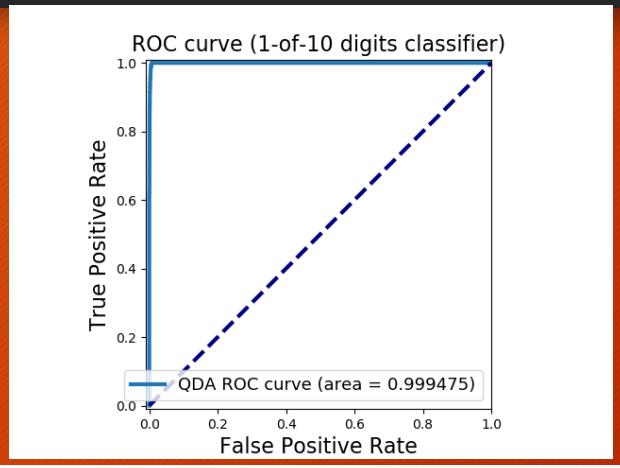
Confusion Matrix

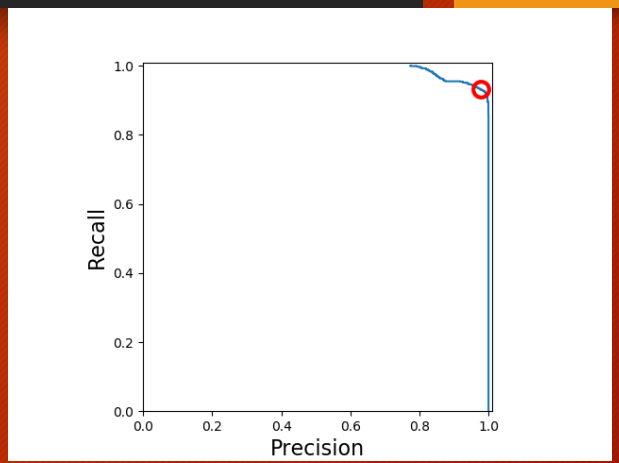
Linear Discriminant Analysis (LDA)

Measure	Value	Derivations
Sensitivity	0.5708	TPR = TP / (TP + FN)
Specificity	0.9993	SPC = TN / (FP + TN)
Precision	0.9247	PPV = TP / (TP + FP)
Negative Predictive Value	0.9939	NPV = TN / (TN + FN)
False Positive Rate	0.0007	FPR = FP / (FP + TN)
False Discovery Rate	0.0753	FDR = FP / (FP + TP)
False Negative Rate	0.4292	FNR = FN / (FN + TP)
Accuracy	0.9933	ACC = (TP + TN) / (P + N)
F1 Score	0.7058	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.7237	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Quadratic Discriminant Analysis (QDA)

- ROC curve
- Recall vs. Precision
- Confusion Matrix






```
In [5]: from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
        from sklearn.metrics import confusion_matrix

        qda = QuadraticDiscriminantAnalysis().fit(X_train, y_train)
        qda_predicted = qda.predict(X_test)
        confusion = confusion_matrix(y_test, qda_predicted)

        print('Quadratic Discriminant Analysis classifier (default settings)\n', confusion)
```

```
Quadratic Discriminant Analysis classifier (default settings)
 [[309626   2250]
 [     1   2696]]
```

Confusion Matrix

Quadratic Discriminant Analysis (QDA) model performance

Measure	Value	Derivations
Sensitivity	0.5451	TPR = TP / (TP + FN)
Specificity	1.0000	SPC = TN / (FP + TN)
Precision	0.9996	PPV = TP / (TP + FP)
Negative Predictive Value	0.9928	NPV = TN / (TN + FN)
False Positive Rate	0.0000	FPR = FP / (FP + TN)
False Discovery Rate	0.0004	FDR = FP / (FP + TP)
False Negative Rate	0.4549	FNR = FN / (FN + TP)
Accuracy	0.9928	ACC = (TP + TN) / (P + N)
F1 Score	0.7055	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.7355	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Logistic Regression

- Confusion Matrix

In [10]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

lr = LogisticRegression().fit(X_train, y_train)
lr_predicted = lr.predict(X_test)
confusion = confusion_matrix(y_test, lr_predicted)

print('Logistic regression classifier (default settings)\n', confusion)
```

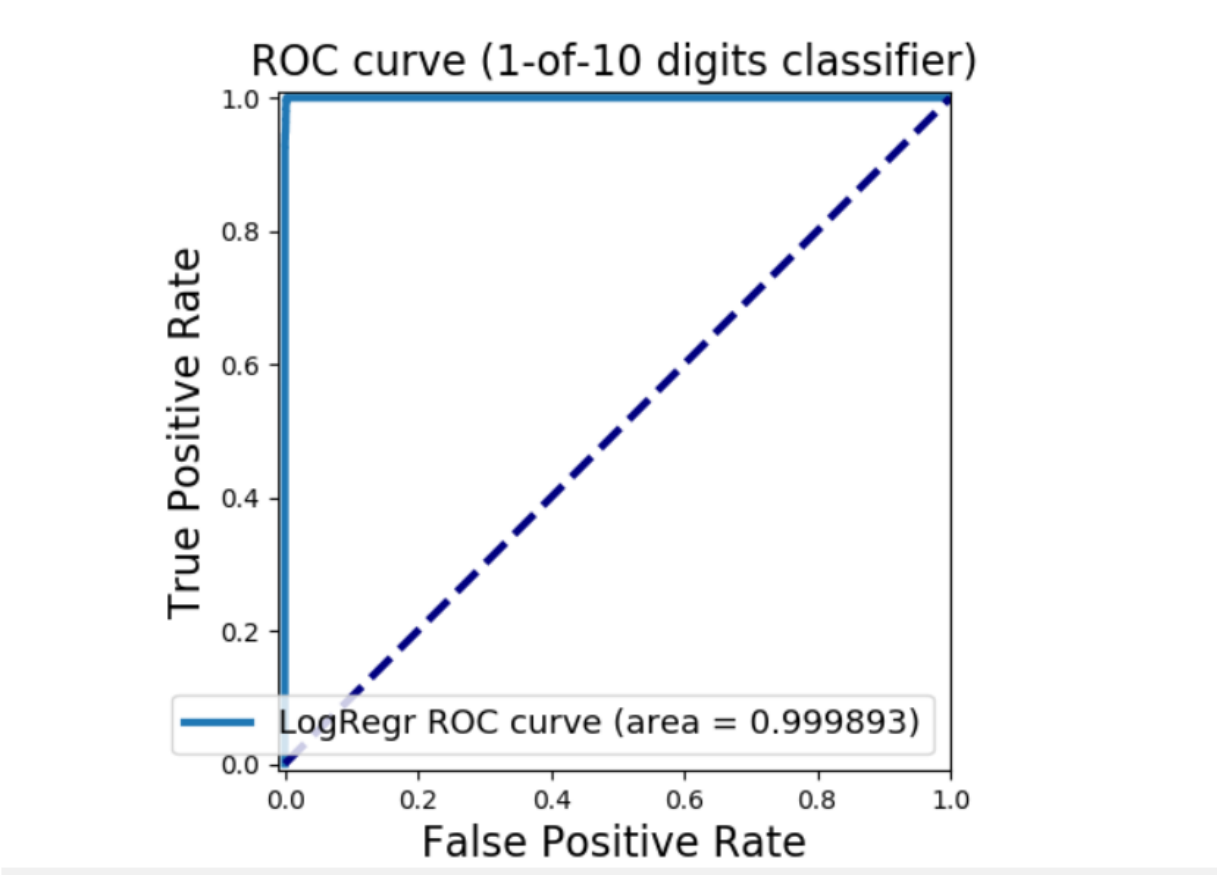
```
C:\Users\gladi\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default so
lver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

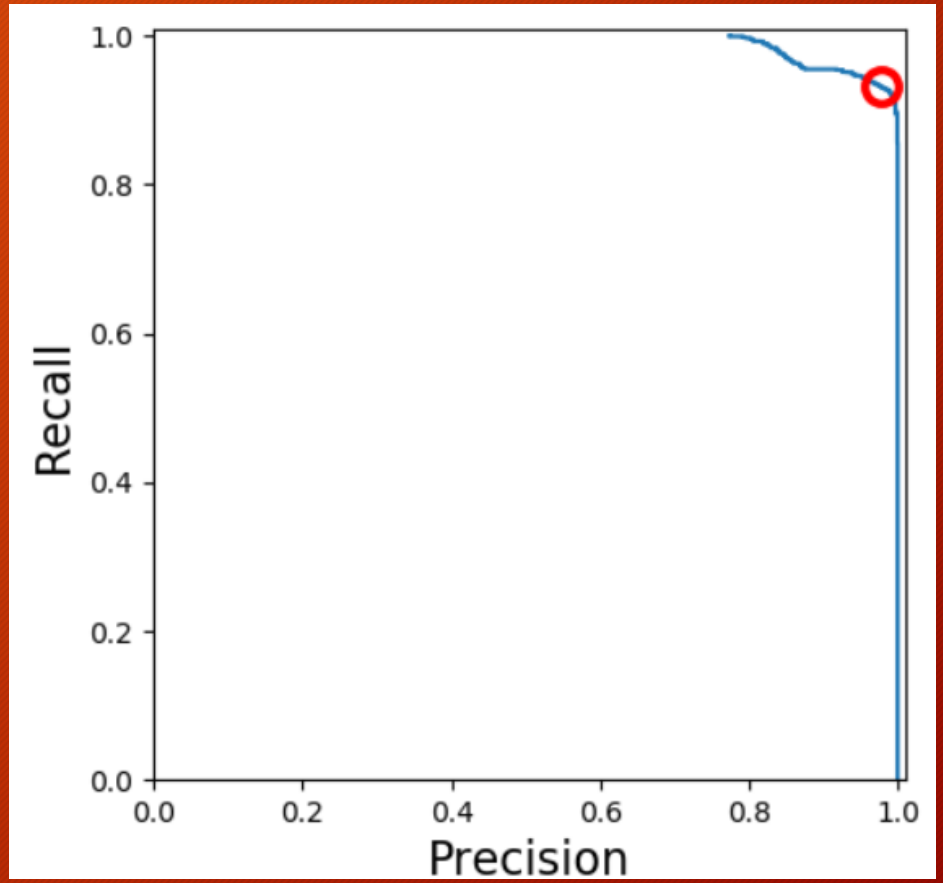
```
Logistic regression classifier (default settings)
 [[311777     56]
 [   205   2535]]
```

Confusion Matrix

Logistic Regression

- ROC curve of logistic regression





Logistic Regression model performance

Measure	Value	Derivations
Sensitivity	0.9784	TPR = TP / (TP + FN)
Specificity	0.9993	SPC = TN / (FP + TN)
Precision	0.9252	PPV = TP / (TP + FP)
Negative Predictive Value	0.9998	NPV = TN / (TN + FN)
False Positive Rate	0.0007	FPR = FP / (FP + TN)
False Discovery Rate	0.0748	FDR = FP / (FP + TP)
False Negative Rate	0.0216	FNR = FN / (FN + TP)
Accuracy	0.9992	ACC = (TP + TN) / (P + N)
F1 Score	0.9510	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.9510	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Random Forest

```
In [13]: print("=== Confusion Matrix ===")
         print(confusion_matrix(y_test, rfc_predict))
         print('\n')
         print("=== Classification Report ===")
         print(classification_report(y_test, rfc_predict))
         print('\n')
         print("=== All AUC Scores ===")
         print(rfc_cv_score)
         print('\n')
         print("=== Mean AUC Score ===")
         print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
```

```
=== Confusion Matrix ===
[[  2668     13]
 [    30 311862]]
```

Confusion Matrix

```
=== Classification Report ===
              precision    recall  f1-score   support

        blue       0.99      1.00      0.99      2681
     nonblue       1.00      1.00      1.00    311892

    accuracy                           1.00    314573
   macro avg       0.99      1.00      1.00    314573
weighted avg       1.00      1.00      1.00    314573
```

- Tree size = 100
- Selected predictors = 2

Random Forest model performance

Measure	Value	Derivations
Sensitivity	0.9889	TPR = TP / (TP + FN)
Specificity	1.0000	SPC = TN / (FP + TN)
Precision	0.9952	PPV = TP / (TP + FP)
Negative Predictive Value	0.9999	NPV = TN / (TN + FN)
False Positive Rate	0.0000	FPR = FP / (FP + TN)
False Discovery Rate	0.0048	FDR = FP / (FP + TP)
False Negative Rate	0.0111	FNR = FN / (FN + TP)
Accuracy	0.9999	ACC = (TP + TN) / (P + N)
F1 Score	0.9920	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.9919	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Support-vector machine (SVM)

- Kernel = linear

In [22]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, precision_recall_f

clf = SVC(kernel = 'linear').fit(x_train,y_train)
clf.predict(x_train)
y_pred = clf.predict(x_test)

# Creates a confusion matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[22]:
```
array([[  2404,    237],
       [    18, 311914]], dtype=int64)
```

Confusion Matrix

Support-vector machine (SVM) model performance

Measure	Value	Derivations
Sensitivity	0.9926	TPR = TP / (TP + FN)
Specificity	0.9992	SPC = TN / (FP + TN)
Precision	0.9103	PPV = TP / (TP + FP)
Negative Predictive Value	0.9999	NPV = TN / (TN + FN)
False Positive Rate	0.0008	FPR = FP / (FP + TN)
False Discovery Rate	0.0897	FDR = FP / (FP + TP)
False Negative Rate	0.0074	FNR = FN / (FN + TP)
Accuracy	0.9992	ACC = (TP + TN) / (P + N)
F1 Score	0.9496	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.9501	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))