Wine Quality data test

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score


df = pd.read_csv('Wine_Quality_Data.csv')
df.head(10)
```

```
   fixed_acidity  volatile_acidity  citric_acid  residual_sugar
chlorides  \
0            7.4              0.70         0.00             1.9
0.076
1            7.8              0.88         0.00             2.6
0.098
2            7.8              0.76         0.04             2.3
0.092
3           11.2              0.28         0.56             1.9
0.075
4            7.4              0.70         0.00             1.9
0.076
5            7.4              0.66         0.00             1.8
0.075
6            7.9              0.60         0.06             1.6
0.069
7            7.3              0.65         0.00             1.2
0.065
8            7.8              0.58         0.02             2.0
0.073
9            7.5              0.50         0.36             6.1
0.071

   free_sulfur_dioxide  total_sulfur_dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56

1                 25.0                  67.0   0.9968  3.20       0.68
```

```
2                  15.0                    54.0   0.9970  3.26       0.65

3                  17.0                    60.0   0.9980  3.16       0.58

4                  11.0                    34.0   0.9978  3.51       0.56

5                  13.0                    40.0   0.9978  3.51       0.56

6                  15.0                    59.0   0.9964  3.30       0.46

7                  15.0                    21.0   0.9946  3.39       0.47

8                   9.0                    18.0   0.9968  3.36       0.57

9                  17.0                   102.0   0.9978  3.35       0.80
```

```
   alcohol  quality color
0      9.4        5   red
1      9.8        5   red
2      9.8        5   red
3      9.8        6   red
4      9.4        5   red
5      9.4        5   red
6      9.4        5   red
7     10.0        7   red
8      9.5        7   red
9     10.5        5   red
```

```
df['color'].value_counts()
```

```
white    4898
red      1599
Name: color, dtype: int64
```

```
X=df[['fixed_acidity',
 'volatile_acidity',
 'citric_acid',
 'residual_sugar',
 'chlorides',
 'free_sulfur_dioxide',
 'total_sulfur_dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol',
 'quality']].values
X[0:5]
```

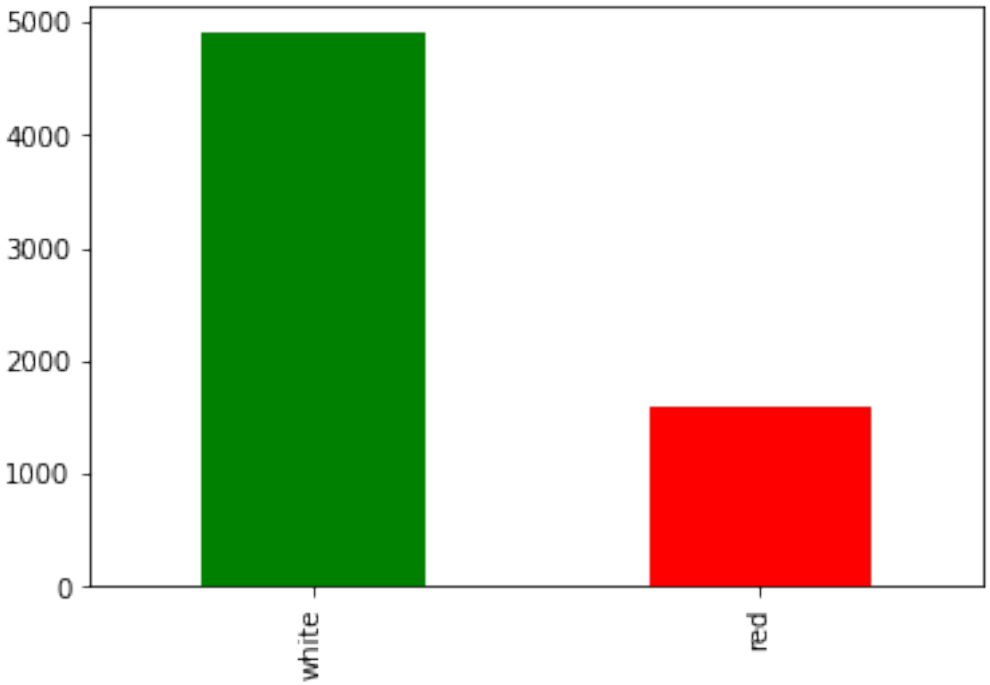
```
y=df[['color']].values
y[0:5]
```

```
array([['red'],
       ['red'],
       ['red'],
       ['red'],
       ['red']], dtype=object)
```

```
from sklearn import preprocessing
le_color = preprocessing.LabelEncoder()
le_color.fit(['white','red'])
y[:,-1] = le_color.transform(y[:,-1])
```

```
df['color'].value_counts().plot.bar(color=['green', 'red'])
```

```
<AxesSubplot:>
```




```
Y = df.color
X = df.drop('color', axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.2, random_state = 0)
```

```
def models(X_train,Y_train):

  #Using Logistic Regression Algorithm to the Training Set
  from sklearn.linear_model import LogisticRegression
  log = LogisticRegression(random_state = 0)
  log.fit(X_train, Y_train)
```

```
  #Using KNeighborsClassifier Method of neighbors class to use Nearest
Neighbor algorithm
  from sklearn.neighbors import KNeighborsClassifier
  knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p
= 2)
  knn.fit(X_train, Y_train)

  #Using SVC method of svm class to use Support Vector Machine
Algorithm
  from sklearn.svm import SVC
  svc_lin = SVC(kernel = 'linear', random_state = 0)
  svc_lin.fit(X_train, Y_train)

  #Using SVC method of svm class to use Kernel SVM Algorithm
  from sklearn.svm import SVC
  svc_rbf = SVC(kernel = 'rbf', random_state = 0)
  svc_rbf.fit(X_train, Y_train)

  #Using GaussianNB method of naïve_bayes class to use Naïve Bayes
Algorithm
  from sklearn.naive_bayes import GaussianNB
  gauss = GaussianNB()
  gauss.fit(X_train, Y_train)

  #Using DecisionTreeClassifier of tree class to use Decision Tree
Algorithm
  from sklearn.tree import DecisionTreeClassifier
  tree = DecisionTreeClassifier(criterion = 'entropy', random_state =
0)
  tree.fit(X_train, Y_train)

  #Using RandomForestClassifier method of ensemble class to use Random
Forest Classification algorithm
  from sklearn.ensemble import RandomForestClassifier
  forest = RandomForestClassifier(n_estimators = 10, criterion =
'entropy', random_state = 0)
  forest.fit(X_train, Y_train)

  #print model accuracy on the training data.
  print('[0]Logistic Regression Training Accuracy:',
log.score(X_train, Y_train))
  print('[1]K Nearest Neighbor Training Accuracy:', knn.score(X_train,
Y_train))
  print('[2]Support Vector Machine (Linear Classifier) Training
Accuracy:', svc_lin.score(X_train, Y_train))
  print('[3]Support Vector Machine (RBF Classifier) Training
Accuracy:', svc_rbf.score(X_train, Y_train))
  print('[4]Gaussian Naive Bayes Training Accuracy:',
gauss.score(X_train, Y_train))
```

```
  print('[5]Decision Tree Classifier Training Accuracy:',
tree.score(X_train, Y_train))
  print('[6]Random Forest Classifier Training Accuracy:',
forest.score(X_train, Y_train))

  return log, knn, svc_lin, svc_rbf, gauss, tree, forest

model = models(X_train,y_train)
```

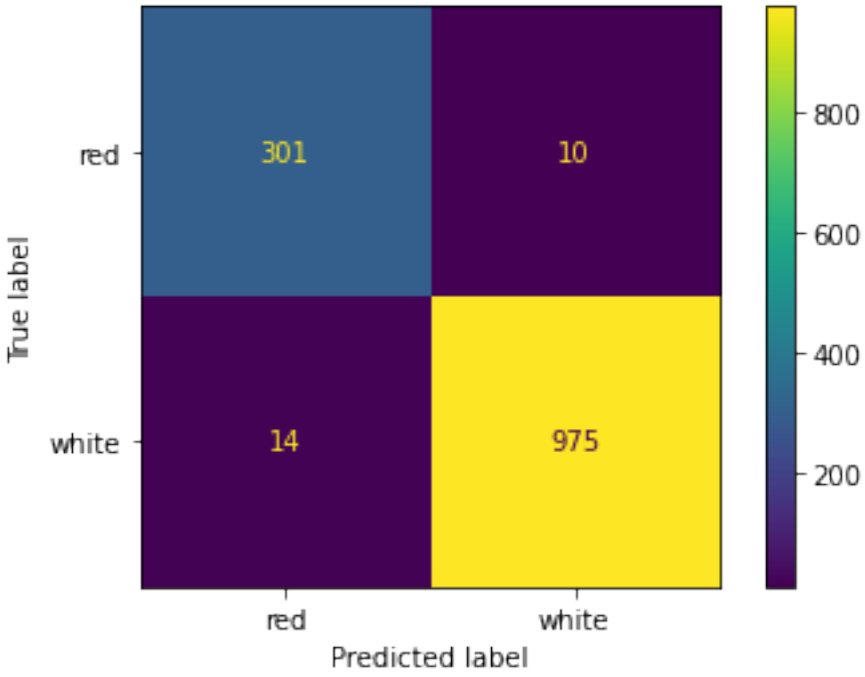
```
c:\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:814:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

[0]Logistic Regression Training Accuracy: 0.9790263613623245
[1]K Nearest Neighbor Training Accuracy: 0.9570906292091591
[2]Support Vector Machine (Linear Classifier) Training Accuracy:
0.9878776217048297
[3]Support Vector Machine (RBF Classifier) Training Accuracy:
0.9363094092745815
[4]Gaussian Naive Bayes Training Accuracy: 0.9697902636136232
[5]Decision Tree Classifier Training Accuracy: 1.0
[6]Random Forest Classifier Training Accuracy: 0.9998075812969021
```

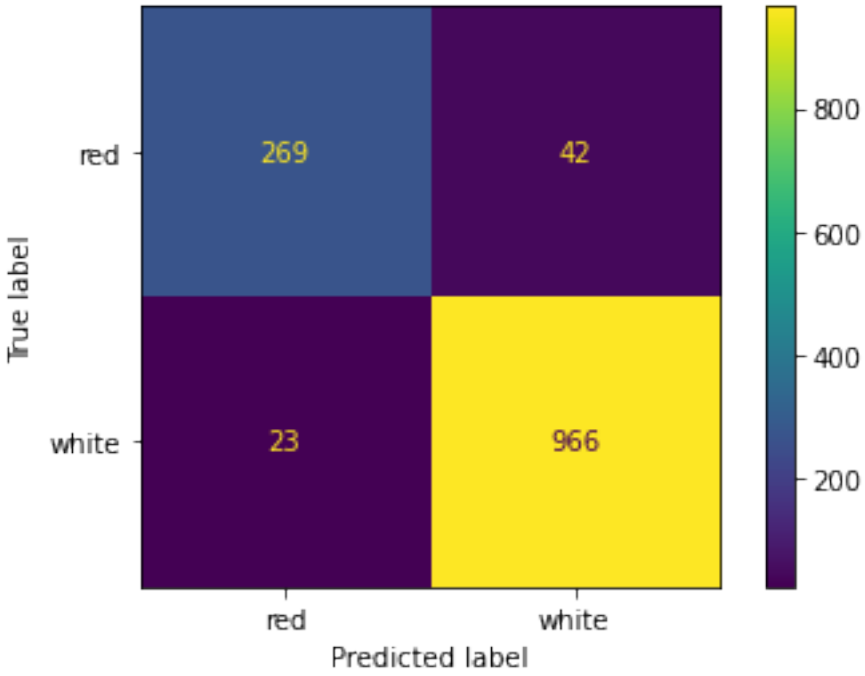
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
l1=list()
accuracy_scores = list()
precision_scores = list()
for i in range(len(model)):
   cm = confusion_matrix(y_test, model[i].predict(X_test))
   #extracting TN, FP, FN, TP
   TN, FP, FN, TP = confusion_matrix(y_test,
model[i].predict(X_test)).ravel()
   print(cm)
   print('Model[{}] Testing Accuracy = "{} !"'.format(i,  (TP + TN) /
(TP + TN + FN + FP)))
   acc=(TP + TN) / (TP + TN + FN + FP)
   print('Model[{}] Testing Precision = "{} !"'.format(i,  (TP ) / (TP
+ FP)))
   prec= (TP) / (TP +  FP)
   print()# Print a new line
   plot_confusion_matrix(model[i], X_test, y_test)
```

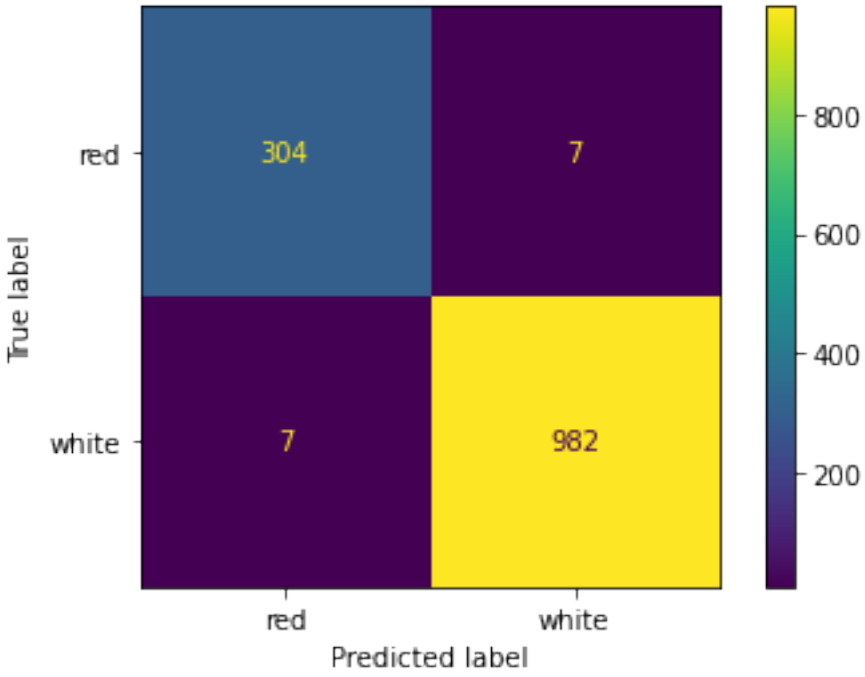
```
   accuracy_scores.append(acc)
   precision_scores.append(prec)
   l1.append(pd.Series({'model': model[i],
'accuracy':acc,'precision':prec }))
```

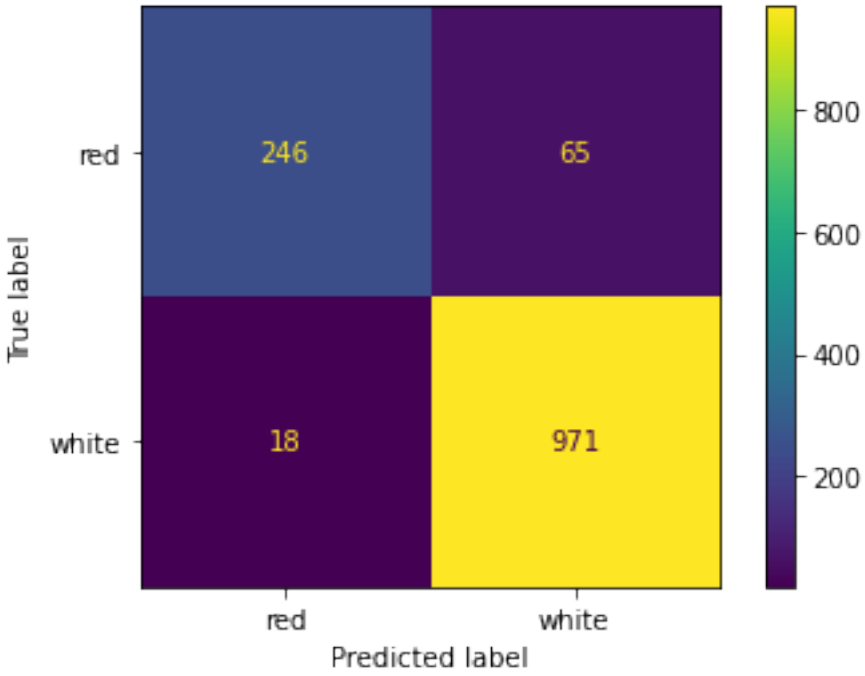
```
[[301  10]
 [ 14 975]]
Model[0] Testing Accuracy = "0.9815384615384616 !"
Model[0] Testing Precision = "0.9898477157360406 !"


c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)

[[269  42]
 [ 23 966]]
Model[1] Testing Accuracy = "0.95 !"
Model[1] Testing Precision = "0.9583333333333334 !"

[[304   7]
 [  7 982]]
Model[2] Testing Accuracy = "0.9892307692307692 !"
Model[2] Testing Precision = "0.9929221435793731 !"


c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)

[[246  65]
 [ 18 971]]
Model[3] Testing Accuracy = "0.9361538461538461 !"
Model[3] Testing Precision = "0.9372586872586872 !"

[[302   9]
```

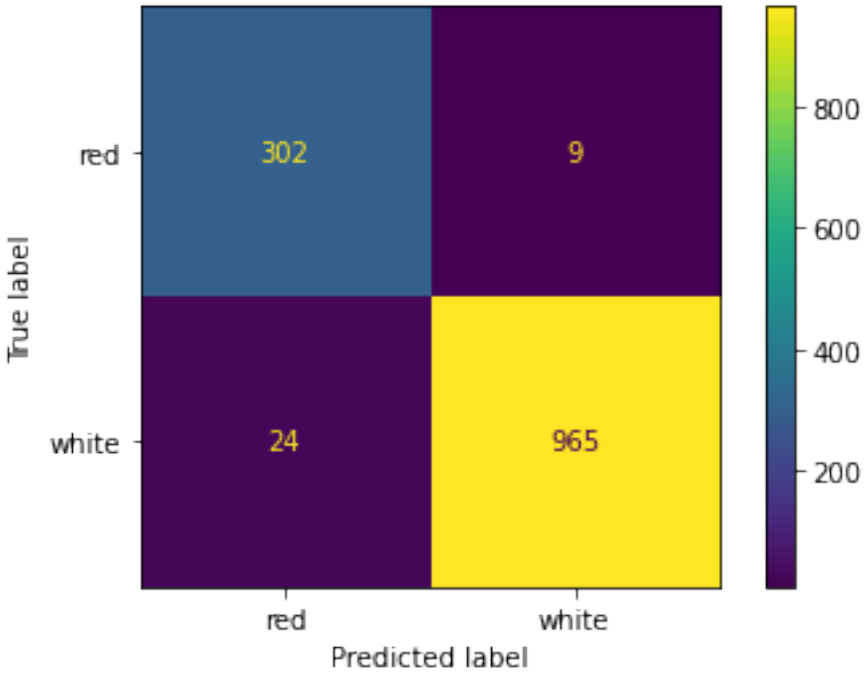
```
 [ 24 965]]
Model[4] Testing Accuracy = "0.9746153846153847 !"
Model[4] Testing Precision = "0.9907597535934292 !"


c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)

[[307   4]
 [ 13 976]]
Model[5] Testing Accuracy = "0.9869230769230769 !"
Model[5] Testing Precision = "0.9959183673469387 !"

[[310   1]
 [  5 984]]
Model[6] Testing Accuracy = "0.9953846153846154 !"
Model[6] Testing Precision = "0.9989847715736041 !"


c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in
1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

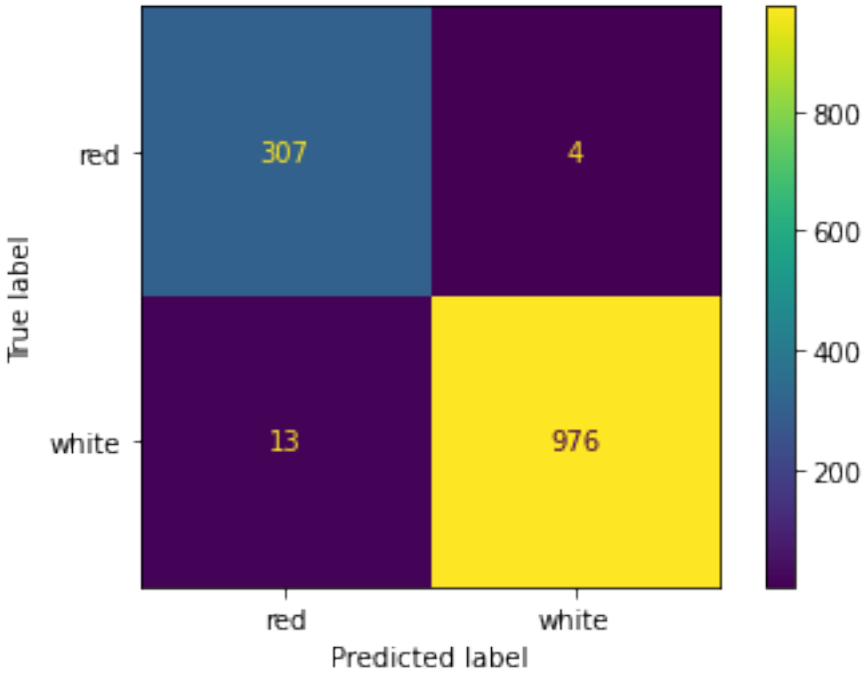

301
10
14
975
red
white
True label
red
white
Predicted label
800
600
400
200

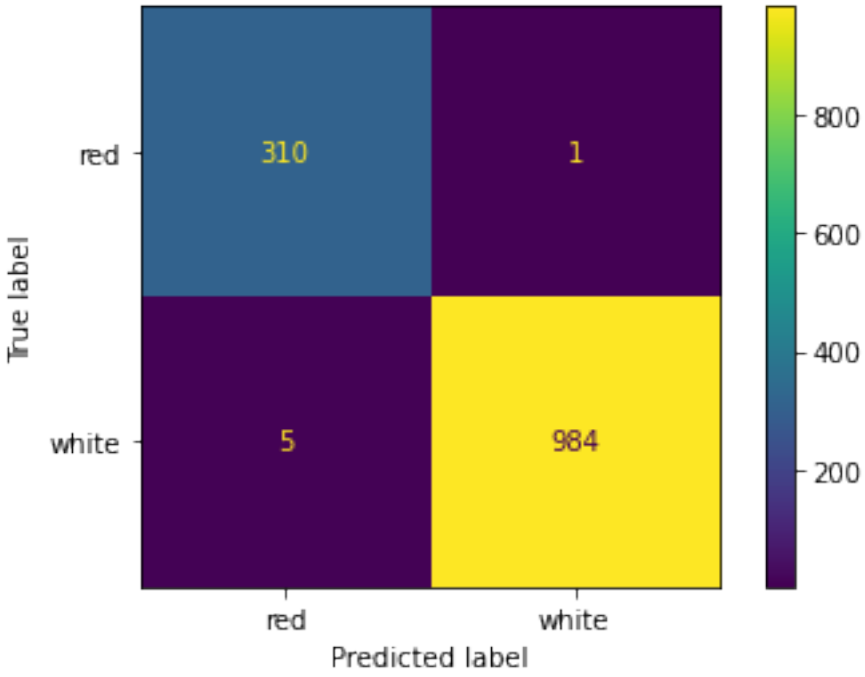

269
42
23
966
red
white
True label
red
white
Predicted label
800
600
400
200


304
7
7
982
red
white
True label
red
white
Predicted label
800
600
400
200


246
65
18
971
red
white
True label
red
white
Predicted label
800
600
400
200


302
9
24
965
red
white
True label
red
white
Predicted label
800
600
400
200


307
4
13
976
red
white
True label
red
white
Predicted label
800
600
400
200




```
models=['LogisticRegression','Knn','SVC_Linear','SVC_RBF','GaussianNB'
,'DecisionTree','RandomForest']

performance_df =
pd.DataFrame({'Algorithm':models,'Accuracy':accuracy_scores,'Precision
':precision_scores}).sort_values('Precision',ascending=False)

performance_df
```

	Algorithm	Accuracy	Precision
6	RandomForest	0.995385	0.998985
5	DecisionTree	0.986923	0.995918
2	SVC_Linear	0.989231	0.992922
4	GaussianNB	0.974615	0.990760
0	LogisticRegression	0.981538	0.989848
1	Knn	0.950000	0.958333
3	SVC_RBF	0.936154	0.937259

```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

	Algorithm	variable	value
0	RandomForest	Accuracy	0.995385
1	DecisionTree	Accuracy	0.986923
2	SVC_Linear	Accuracy	0.989231
3	GaussianNB	Accuracy	0.974615
4	LogisticRegression	Accuracy	0.981538
5	Knn	Accuracy	0.950000
6	SVC_RBF	Accuracy	0.936154
7	RandomForest	Precision	0.998985

```
8          DecisionTree  Precision  0.995918
9            SVC_Linear  Precision  0.992922
10           GaussianNB  Precision  0.990760
11   LogisticRegression  Precision  0.989848
12                  Knn  Precision  0.958333
13              SVC_RBF  Precision  0.937259
```

```
sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1,
kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```