THE MET LEAGUE OF COLLEGES
MET Bhujbal Knowledge City
AS SHARP AS YOU CAN GET

MET Bhujbal Knowledge City, Nashik

DATA MINING AND WAREHOUSING MINI-PROJECT REPORT

Title - Wine Quality Testing

SUBMITTED BY

Name	Roll No
Abhishek Abhay Palve	**08**
Khushal Deepesh Patel	**45**

Under the guidance of
Prof. Vishal Patil

DEPARTMENT OF COMPUTER ENGINEERING
Academic Year 2021-22

Contents

Problem Statement

Consider a labelled dataset belonging to an application domain. Apply suitable data pre-processing steps such as handling of null values, data reduction, discretization. For prediction of class labels of given data instances, build classifier models using different techniques (minimum 3), analyse the confusion matrix and compare these models. Also apply cross validation while preparing the training and testing datasets.

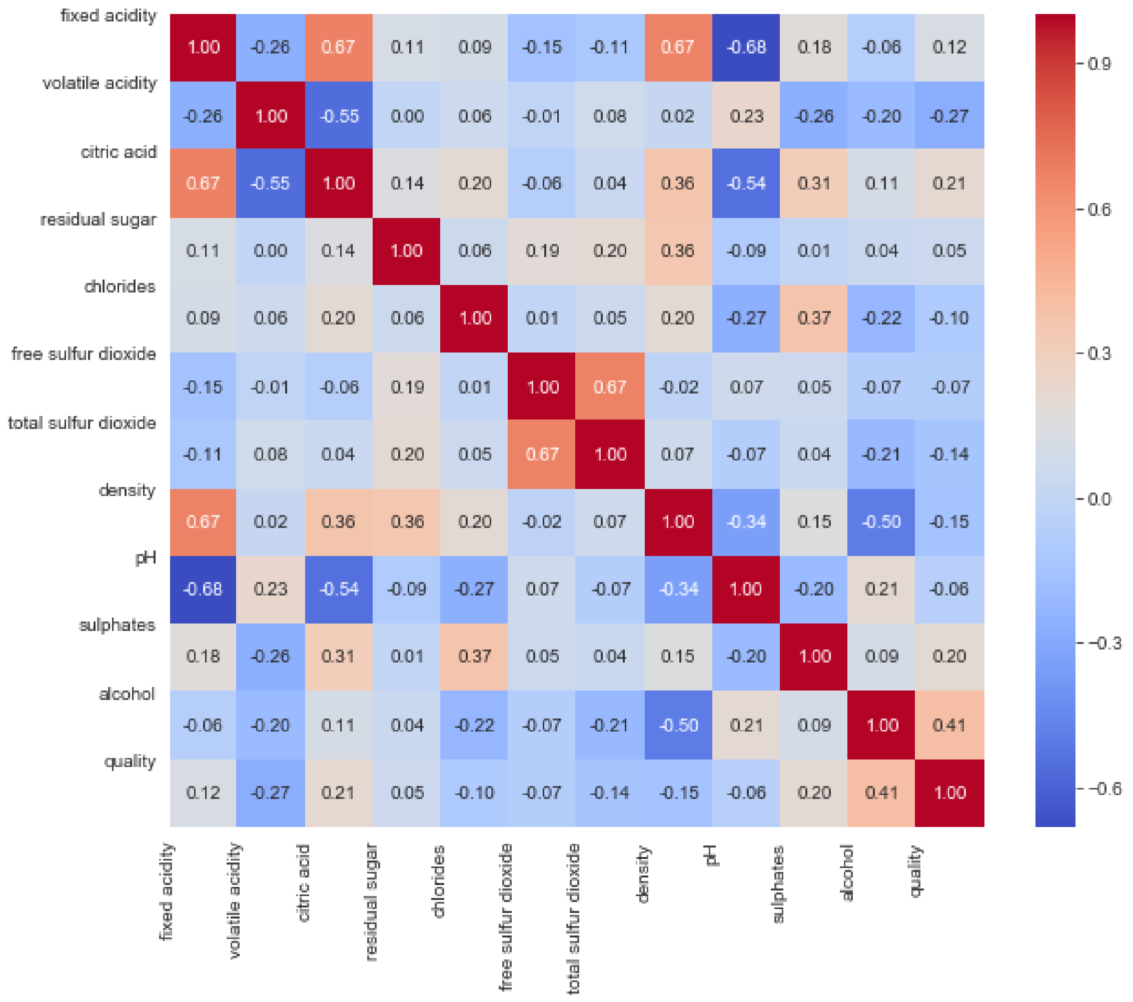
Abstract

Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels. For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. In this project we use multiple classification models to analyse the outcome of hockey game played between various teams. Use apply suitable data pre-processing steps. We then compare performance of classification models to find which one is the best.

Introduction

We have been provided with the data regarding various characteristic of wine, The Data fields are

1. Fixed acidity- Fixed acidity pH in mixture
2. Volatile acidity – Volatile acidity pH in mixture.
3. Citric acid – Citric acid composition in mixture.
4. Residual sugar – Sugar composition in mixture.
5. Chlorides – Chloride composition in total mixture.
6. Free Sulphur dioxide – Sulphur dioxide composition in mixture.
7. Total Sulphur dioxide – Total Sulphur dioxide composition in mixture.
8. Density – Density of mixture.
9. pH – pH of liquid mixture.
10. Sulphates- Sulphates composition in mixture.
11. Alcohol- Total alcohol composition in wine.
12. Quality- Quality index based on given compositions.



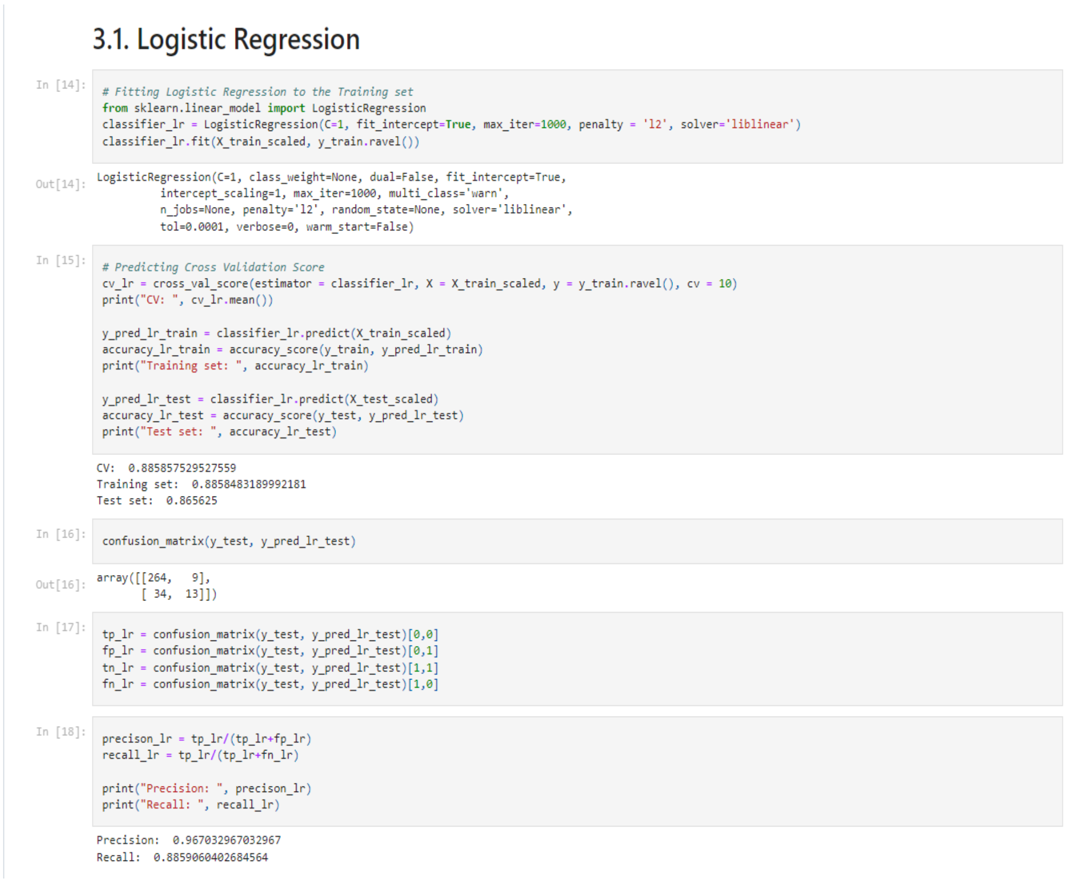
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.00	-0.26	0.67	0.11	0.09	-0.15	-0.11	0.67	-0.68	0.18	-0.06	0.12
volatile acidity	-0.26	1.00	-0.55	0.00	0.06	-0.01	0.08	0.02	0.23	-0.26	-0.20	-0.27
citric acid	0.67	-0.55	1.00	0.14	0.20	-0.06	0.04	0.36	-0.54	0.31	0.11	0.21
residual sugar	0.11	0.00	0.14	1.00	0.06	0.19	0.20	0.36	-0.09	0.01	0.04	0.05
chlorides	0.09	0.06	0.20	0.06	1.00	0.01	0.05	0.20	-0.27	0.37	-0.22	-0.10
free sulfur dioxide	-0.15	-0.01	-0.06	0.19	0.01	1.00	0.67	-0.02	0.07	0.05	-0.07	-0.07
total sulfur dioxide	-0.11	0.08	0.04	0.20	0.05	0.67	1.00	0.07	-0.07	0.04	-0.21	-0.14
density	0.67	0.02	0.36	0.36	0.20	-0.02	0.07	1.00	-0.34	0.15	-0.50	-0.15
pH	-0.68	0.23	-0.54	-0.09	-0.27	0.07	-0.07	-0.34	1.00	-0.20	0.21	-0.06
sulphates	0.18	-0.26	0.31	0.01	0.37	0.05	0.04	0.15	-0.20	1.00	0.09	0.20
alcohol	-0.06	-0.20	0.11	0.04	-0.22	-0.07	-0.21	-0.50	0.21	0.09	1.00	0.41
quality	0.12	-0.27	0.21	0.05	-0.10	-0.07	-0.14	-0.15	-0.06	0.20	0.41	1.00

Figure 1: Confusion matrix

We have trained using two models Logistic Regression, KNN classifier, Gaussian Naïve Bayes and Random Forest Classifier.

- To understand data pre-processing
- To perform classification on dataset and predict labels for test dataset.

Test Cases



3.1. Logistic Regression

```
In [14]:
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier_lr = LogisticRegression(C=1, fit_intercept=True, max_iter=1000, penalty = 'l2', solver='liblinear')
classifier_lr.fit(X_train_scaled, y_train.ravel())
```

```
Out[14]:
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=1000, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
          tol=0.0001, verbose=0, warm_start=False)
```

```
In [15]:
# Predicting Cross Validation Score
cv_lr = cross_val_score(estimator = classifier_lr, X = X_train_scaled, y = y_train.ravel(), cv = 10)
print("CV: ", cv_lr.mean())

y_pred_lr_train = classifier_lr.predict(X_train_scaled)
accuracy_lr_train = accuracy_score(y_train, y_pred_lr_train)
print("Training set: ", accuracy_lr_train)

y_pred_lr_test = classifier_lr.predict(X_test_scaled)
accuracy_lr_test = accuracy_score(y_test, y_pred_lr_test)
print("Test set: ", accuracy_lr_test)
```

```
CV:  0.885857529527559
Training set:  0.8858483189992181
Test set:  0.865625
```

```
In [16]:
confusion_matrix(y_test, y_pred_lr_test)
```

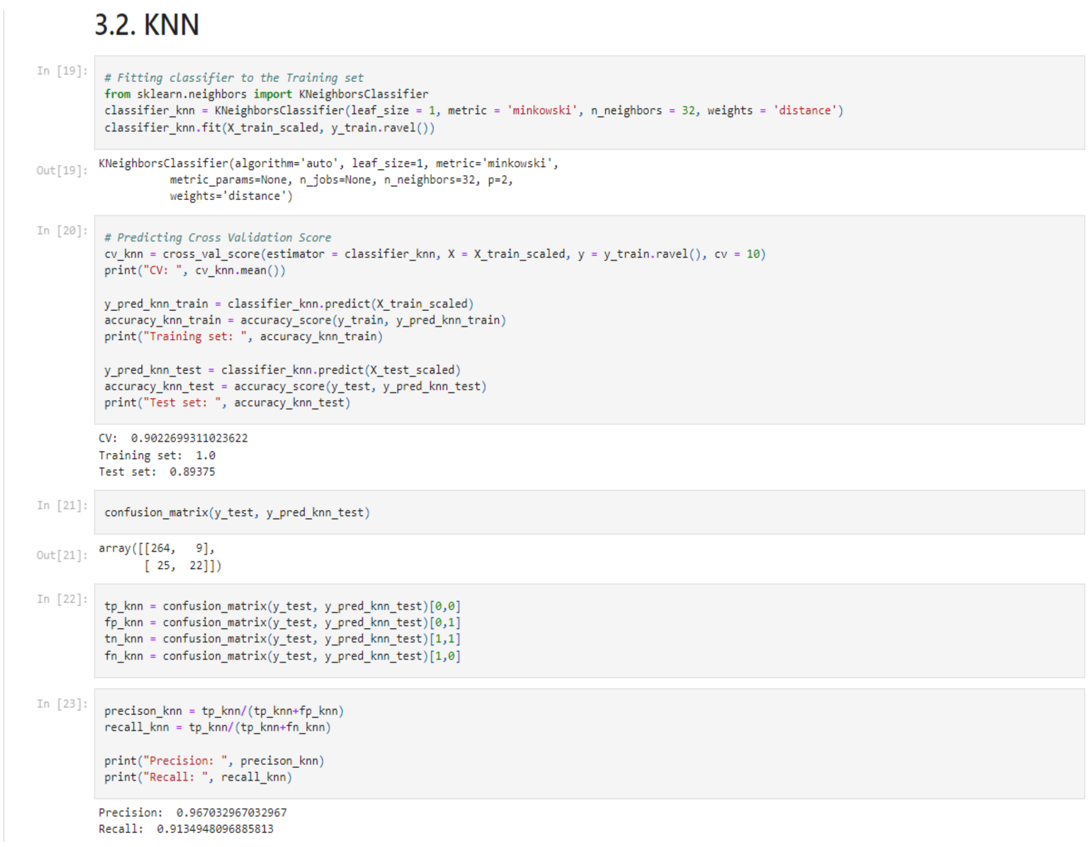
```
Out[16]:
array([[264,   9],
       [ 34,  13]])
```

```
In [17]:
tp_lr = confusion_matrix(y_test, y_pred_lr_test)[0,0]
fp_lr = confusion_matrix(y_test, y_pred_lr_test)[0,1]
tn_lr = confusion_matrix(y_test, y_pred_lr_test)[1,1]
fn_lr = confusion_matrix(y_test, y_pred_lr_test)[1,0]
```

```
In [18]:
precison_lr = tp_lr/(tp_lr+fp_lr)
recall_lr = tp_lr/(tp_lr+fn_lr)

print("Precision: ", precison_lr)
print("Recall: ", recall_lr)
```

```
Precision:  0.967032967032967
Recall:  0.8859060402684564
```

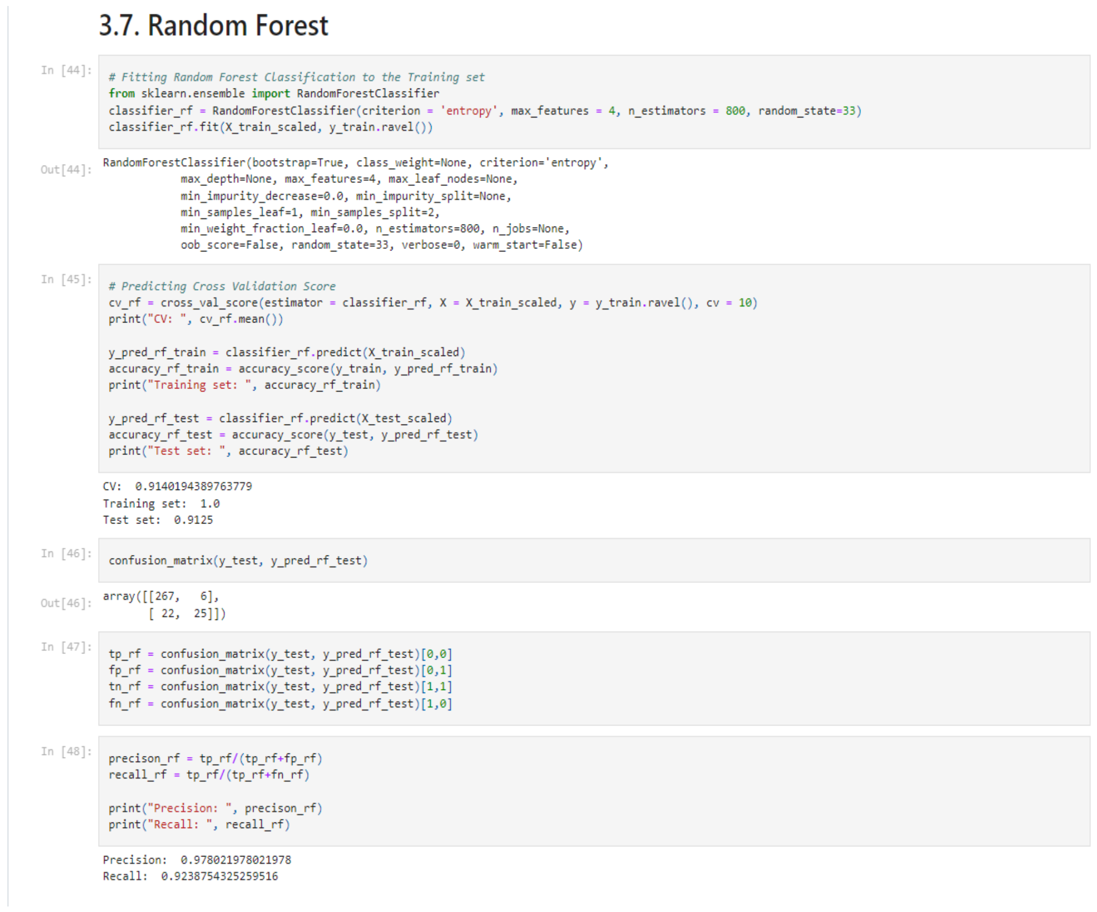
Figure 2: Output for logistic regression



3.2. KNN

```
In [19]: # Fitting classifier to the Training set
         from sklearn.neighbors import KNeighborsClassifier
         classifier_knn = KNeighborsClassifier(leaf_size = 1, metric = 'minkowski', n_neighbors = 32, weights = 'distance')
         classifier_knn.fit(X_train_scaled, y_train.ravel())

Out[19]: KNeighborsClassifier(algorithm='auto', leaf_size=1, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=32, p=2,
                    weights='distance')

In [20]: # Predicting Cross Validation Score
         cv_knn = cross_val_score(estimator = classifier_knn, X = X_train_scaled, y = y_train.ravel(), cv = 10)
         print("CV: ", cv_knn.mean())

         y_pred_knn_train = classifier_knn.predict(X_train_scaled)
         accuracy_knn_train = accuracy_score(y_train, y_pred_knn_train)
         print("Training set: ", accuracy_knn_train)

         y_pred_knn_test = classifier_knn.predict(X_test_scaled)
         accuracy_knn_test = accuracy_score(y_test, y_pred_knn_test)
         print("Test set: ", accuracy_knn_test)

         CV:  0.9022699311023622
         Training set:  1.0
         Test set:  0.89375

In [21]: confusion_matrix(y_test, y_pred_knn_test)

Out[21]: array([[264,   9],
                [ 25,  22]])

In [22]: tp_knn = confusion_matrix(y_test, y_pred_knn_test)[0,0]
         fp_knn = confusion_matrix(y_test, y_pred_knn_test)[0,1]
         tn_knn = confusion_matrix(y_test, y_pred_knn_test)[1,1]
         fn_knn = confusion_matrix(y_test, y_pred_knn_test)[1,0]

In [23]: precison_knn = tp_knn/(tp_knn+fp_knn)
         recall_knn = tp_knn/(tp_knn+fn_knn)

         print("Precision: ", precison_knn)
         print("Recall: ", recall_knn)

         Precision:  0.967032967032967
         Recall:  0.9134948096885813
```

Figure 3: Output for K Neighbours classifier3



3.7. Random Forest

In [44]:
```
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier_rf = RandomForestClassifier(criterion = 'entropy', max_features = 4, n_estimators = 800, random_state=33)
classifier_rf.fit(X_train_scaled, y_train.ravel())
```

Out[44]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features=4, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=800, n_jobs=None,
            oob_score=False, random_state=33, verbose=0, warm_start=False)
```

In [45]:
```
# Predicting Cross Validation Score
cv_rf = cross_val_score(estimator = classifier_rf, X = X_train_scaled, y = y_train.ravel(), cv = 10)
print("CV: ", cv_rf.mean())

y_pred_rf_train = classifier_rf.predict(X_train_scaled)
accuracy_rf_train = accuracy_score(y_train, y_pred_rf_train)
print("Training set: ", accuracy_rf_train)

y_pred_rf_test = classifier_rf.predict(X_test_scaled)
accuracy_rf_test = accuracy_score(y_test, y_pred_rf_test)
print("Test set: ", accuracy_rf_test)
```

```
CV:  0.9140194389763779
Training set:  1.0
Test set:  0.9125
```

In [46]:
```
confusion_matrix(y_test, y_pred_rf_test)
```

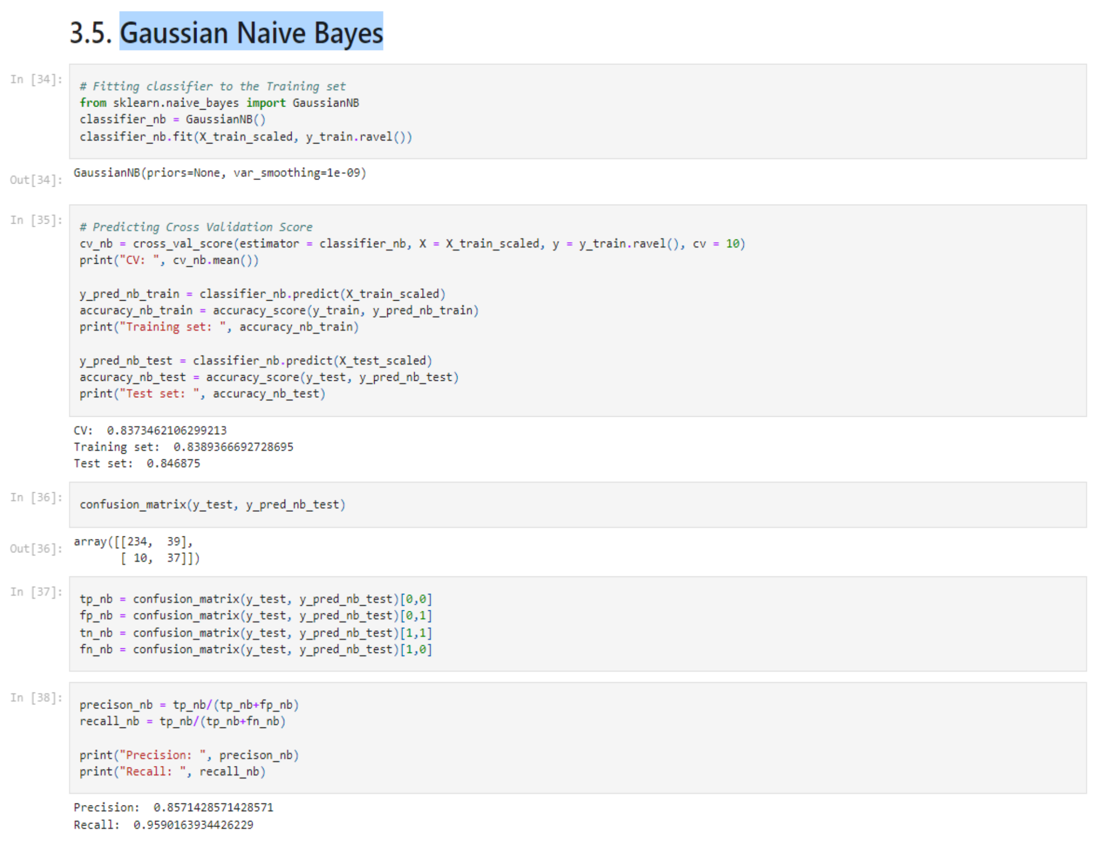
Out[46]:
```
array([[267,   6],
       [ 22,  25]])
```

In [47]:
```
tp_rf = confusion_matrix(y_test, y_pred_rf_test)[0,0]
fp_rf = confusion_matrix(y_test, y_pred_rf_test)[0,1]
tn_rf = confusion_matrix(y_test, y_pred_rf_test)[1,1]
fn_rf = confusion_matrix(y_test, y_pred_rf_test)[1,0]
```

In [48]:
```
precison_rf = tp_rf/(tp_rf+fp_rf)
recall_rf = tp_rf/(tp_rf+fn_rf)

print("Precision: ", precison_rf)
print("Recall: ", recall_rf)
```

```
Precision:  0.978021978021978
Recall:  0.9238754325259516
```

Figure 4: Output for random forest classifier



3.5. Gaussian Naive Bayes

```
In [34]:  # Fitting classifier to the Training set
          from sklearn.naive_bayes import GaussianNB
          classifier_nb = GaussianNB()
          classifier_nb.fit(X_train_scaled, y_train.ravel())

Out[34]:  GaussianNB(priors=None, var_smoothing=1e-09)

In [35]:  # Predicting Cross Validation Score
          cv_nb = cross_val_score(estimator = classifier_nb, X = X_train_scaled, y = y_train.ravel(), cv = 10)
          print("CV: ", cv_nb.mean())

          y_pred_nb_train = classifier_nb.predict(X_train_scaled)
          accuracy_nb_train = accuracy_score(y_train, y_pred_nb_train)
          print("Training set: ", accuracy_nb_train)

          y_pred_nb_test = classifier_nb.predict(X_test_scaled)
          accuracy_nb_test = accuracy_score(y_test, y_pred_nb_test)
          print("Test set: ", accuracy_nb_test)

          CV:  0.8373462106299213
          Training set:  0.8389366692728695
          Test set:  0.846875

In [36]:  confusion_matrix(y_test, y_pred_nb_test)

Out[36]:  array([[234,  39],
                 [ 10,  37]])

In [37]:  tp_nb = confusion_matrix(y_test, y_pred_nb_test)[0,0]
          fp_nb = confusion_matrix(y_test, y_pred_nb_test)[0,1]
          tn_nb = confusion_matrix(y_test, y_pred_nb_test)[1,1]
          fn_nb = confusion_matrix(y_test, y_pred_nb_test)[1,0]

In [38]:  precison_nb = tp_nb/(tp_nb+fp_nb)
          recall_nb = tp_nb/(tp_nb+fn_nb)

          print("Precision: ", precison_nb)
          print("Recall: ", recall_nb)

          Precision:  0.8571428571428571
          Recall:  0.9590163934426229
```

Figure 5: Output for Gaussian Naïve Bayes classifier

Result

The accuracy for Random Forest classifier is around 80%. while that of other models is lesser. The following are plotting of accuracy of various model outputs.

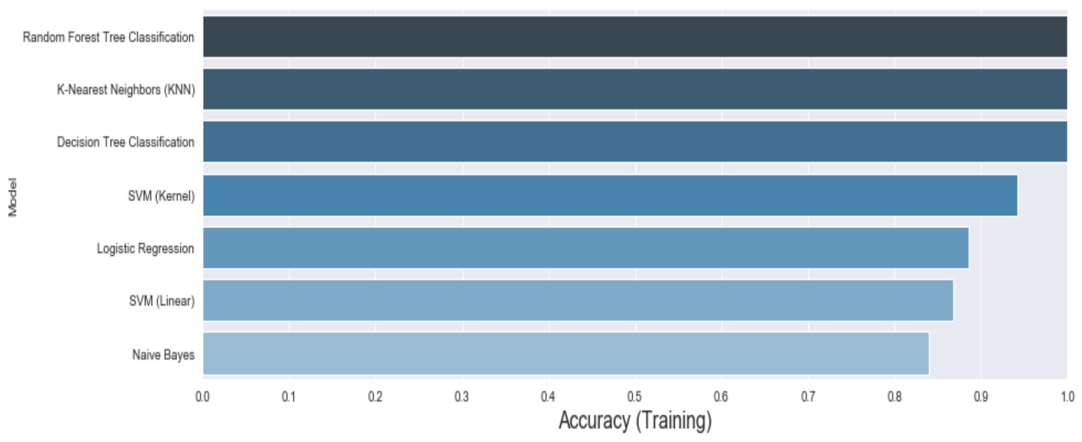
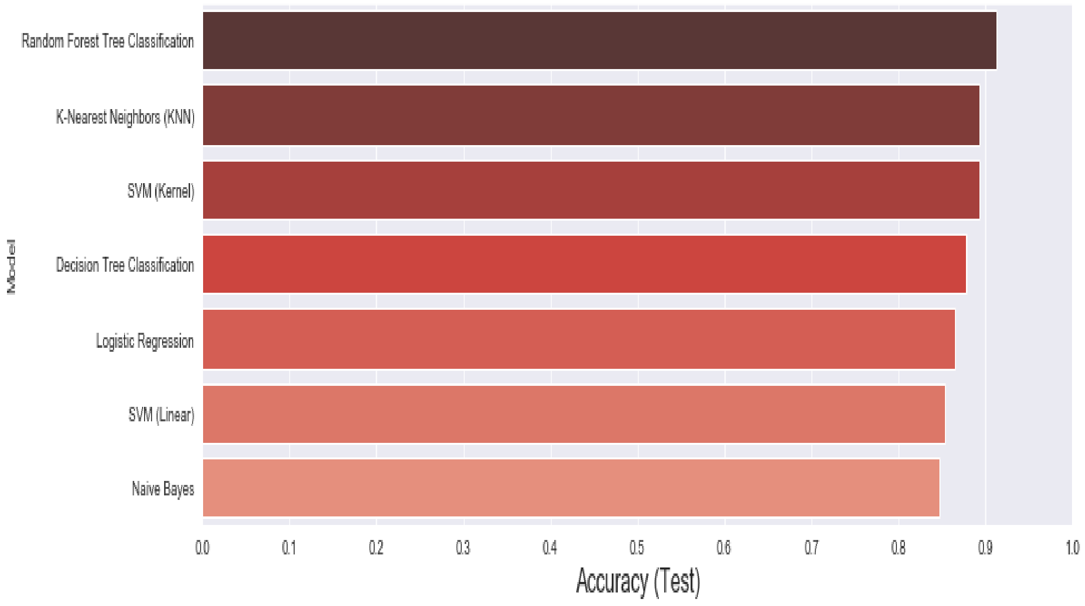


Figure 6: Accuracy result



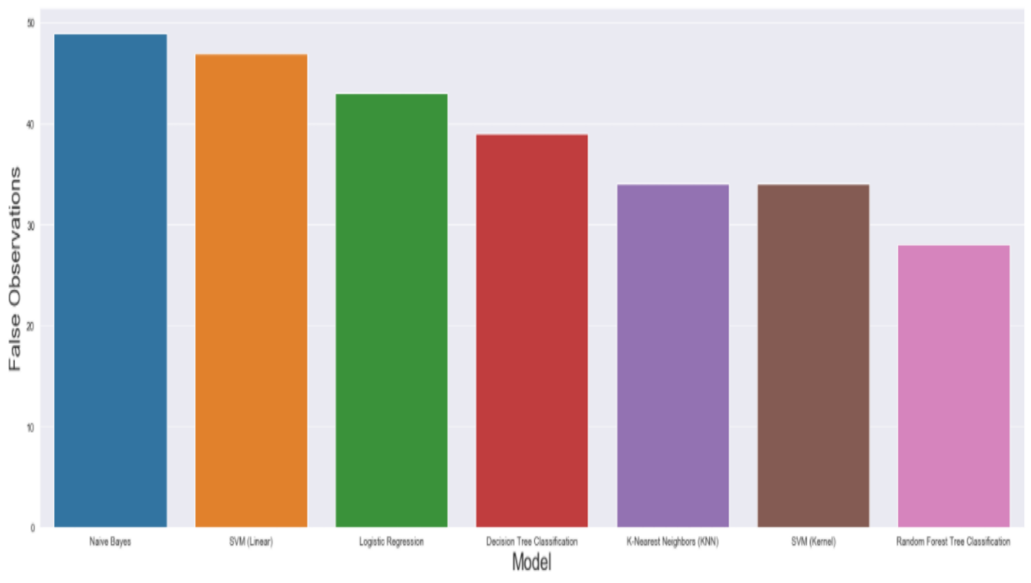


Figure 9: Comparative bar chart of various classifier models

Conclusion

We have analysed the wine quality dataset and performed data pre-processing steps. We have experimented multiple classification models and found out the best performer among them. We have then used this model to make predictions on test dataset.

References

[1] https://www.kaggle.com/c/datawiz19round1/data

[2] https://seaborn.pydata.org/index.html

[3] Jiawei Han, Micheline Kamber, Jian Pei, *Data Mining Concepts and Techniques*3