Water Quality using Novel Supervised Learning Algorithm

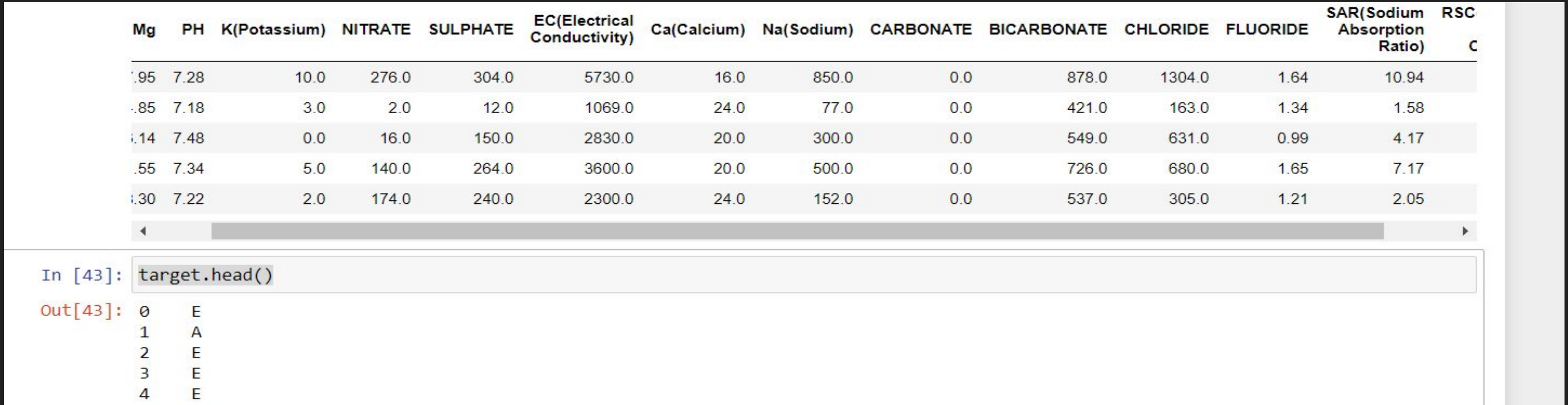
By:
Srikrishna Ganapati Yaji - 181CO153

Dataset Description

1. A dataset with 16 features and 88 rows was given.
2. The feature vectors were:
    a. ID
    b. SiteName
    c. Mg
    d. PH
    e. K
    f. Nitrate
    g. Sulphate
    h. EC
    i. Ca
    j. Na
    k. Carbonate
    l. BiCarbonate
    m. Chloride
    n. Fluoride
    o. SAR
    p. RSC
3. The target was the class to which the water quality belonged to.
4. The first two feature attribute ID and SiteName are useless or redundant as the water quality is independent of the ID and SiteName and depends on the content of various minerals in water.

Dataset Preprocessing

1. The redundant columns of ID and SiteName were removed.
2. The rows and columns with NaN values were removed.
3. The feature vectors and target vectors were separated.
4. The feature vector finally had a shape of: (88 x 14)
5. The target vector finally had a shape of: (88 x 1)



Mg	PH	K(Potassium)	NITRATE	SULPHATE	EC(Electrical Conductivity)	Ca(Calcium)	Na(Sodium)	CARBONATE	BICARBONATE	CHLORIDE	FLUORIDE	SAR(Sodium Absorption Ratio)	RSC C
.95	7.28	10.0	276.0	304.0	5730.0	16.0	850.0	0.0	878.0	1304.0	1.64	10.94	
.85	7.18	3.0	2.0	12.0	1069.0	24.0	77.0	0.0	421.0	163.0	1.34	1.58	
.14	7.48	0.0	16.0	150.0	2830.0	20.0	300.0	0.0	549.0	631.0	0.99	4.17	
.55	7.34	5.0	140.0	264.0	3600.0	20.0	500.0	0.0	726.0	680.0	1.65	7.17	
.30	7.22	2.0	174.0	240.0	2300.0	24.0	152.0	0.0	537.0	305.0	1.21	2.05	

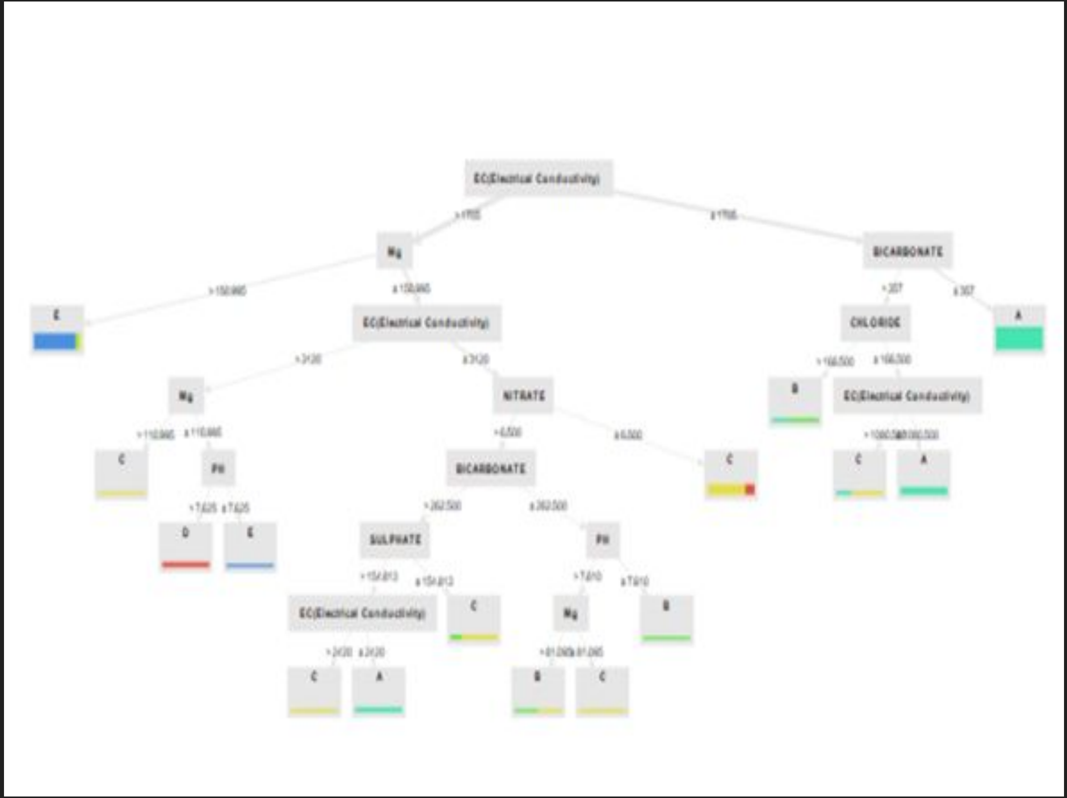
```
In [43]: target.head()

Out[43]: 0    E
         1    A
         2    E
         3    E
         4    E
```

Aim

1. Create various machine learning models learnt and apply them on the given dataset.
2. Get the accuracies of various models and analyse the best model for water quality computation
3. Combine various models learnt in machine learning via ensembling techniques to increase accuracies.
4. Then finally analyse various models created and derive conclusions.

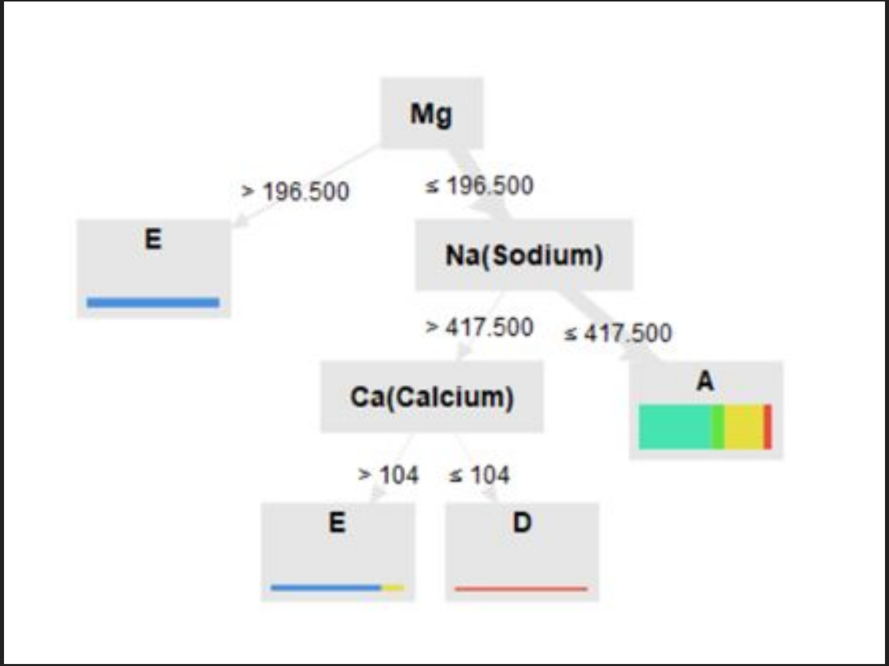
Decision Tree ID3 Algorithm

- Rapid Miner was used for the implementation of the Decision Tree ID3 Algorithm
- The data set was imported, cross validation was applied and finally the model was created and accuracy was computed.
- The accuracy of the decision tree ID3 algorithm was 62.64%


EC(Electrical Conductivity)
Mg
BICARBONATE
E
EC(Electrical Conductivity)
CHLORIDE
A
Mg
NITRATE
B
EC(Electrical Conductivity)
C
PH
BICARBONATE
C
C
A
D
E
SULPHATE
PH
EC(Electrical Conductivity)
C
Mg
B
C
A
B
C

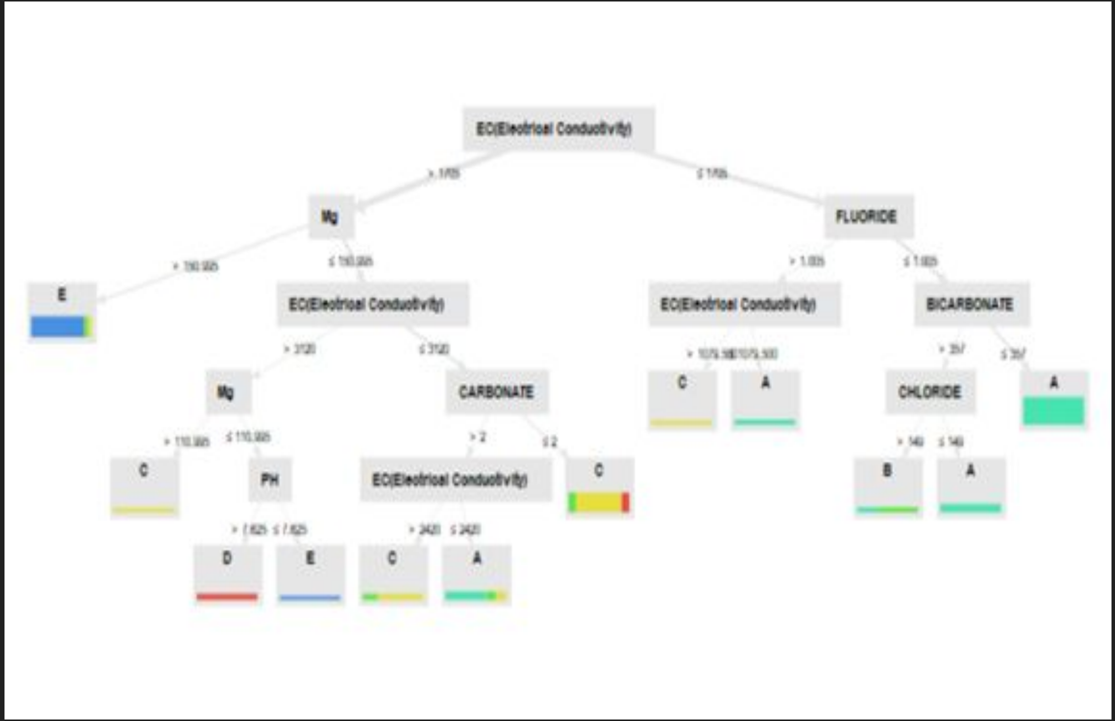
Decision Tree Gain Ratio Algorithm

- Rapid Miner was used for the implementation of the Decision Tree Gain Ratio Algorithm
- The data set was imported, cross validation was applied and finally the model was created and accuracy was computed.
- The accuracy obtained from Gain Ratio Algorithm was 61.39%



Decision Tree Gini Index Algorithm

- Rapid Miner was used for the implementation of the Decision Tree Gini Index Algorithm
- The data set was imported, cross validation was applied and finally the model was created and accuracy was computed.
- The accuracy obtained from Gini Index Algorithm was 62.64%

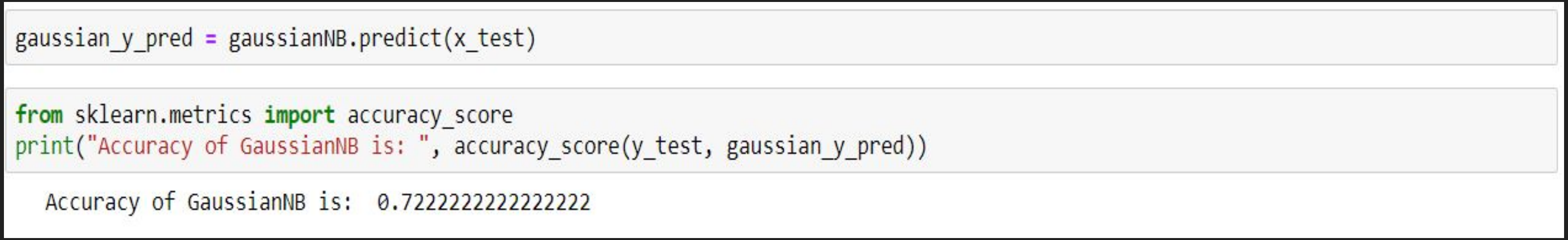


Using Sklearn

- The given water quality data was split into training and testing data in the ratio of 4:1.
- Various models were implemented using sklearn and their accuracy scores were computed.
- The training features and target was named as X_train and Y_train respectively.
- The testing features and target was named as X_test and Y_test respectively

Naive Bayesian Algorithm

- A naive bayesian model was developed using the GaussianNB class in sklearn library.
- It was trained on the X_train and Y_train and tested on the X_test and Y_test part of the dataset.
- The accuracy of the Naive Gaussian Model developed was obtained to be 72.222%



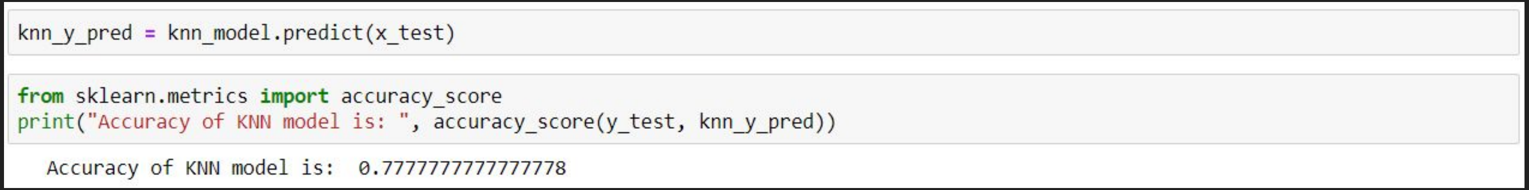
```python
gaussian_y_pred = gaussianNB.predict(x_test)
```

```python
from sklearn.metrics import accuracy_score
print("Accuracy of GaussianNB is: ", accuracy_score(y_test, gaussian_y_pred))
```

```
Accuracy of GaussianNB is:  0.7222222222222222
```

K-Nearest Neighbors Algorithm

- A knn model was developed using the KNeighborsClassfier algorithm using the sklearn library.
- It was trained on the X_train and Y_train and tested on the X_test and Y_test part of the dataset.
- The accuracy of the KNeighborsClassifier model developed was obtained to be 77.777%



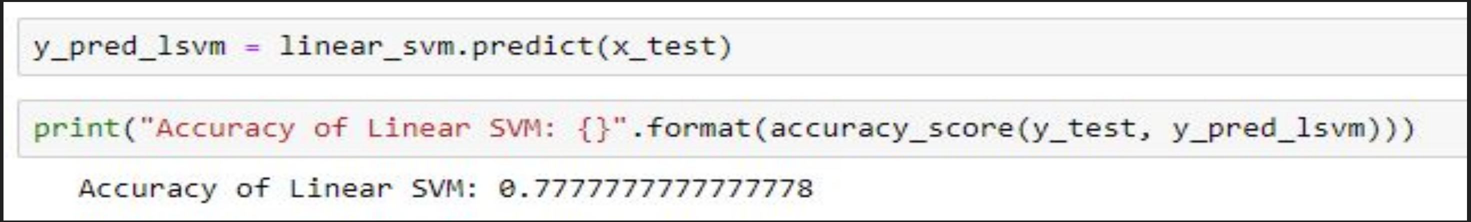
```
knn_y_pred = knn_model.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
print("Accuracy of KNN model is: ", accuracy_score(y_test, knn_y_pred))
```

```
   Accuracy of KNN model is:  0.7777777777777778
```

Linear SVM

- A linear svm model was created using the sklearn library.
- The kernel function was chosen to be identity function.
- It was trained on the X_train and Y_train and tested on the X_test and Y_test part of the dataset.
- The accuracy of the Linear SVM model developed was obtained to be 77.777%



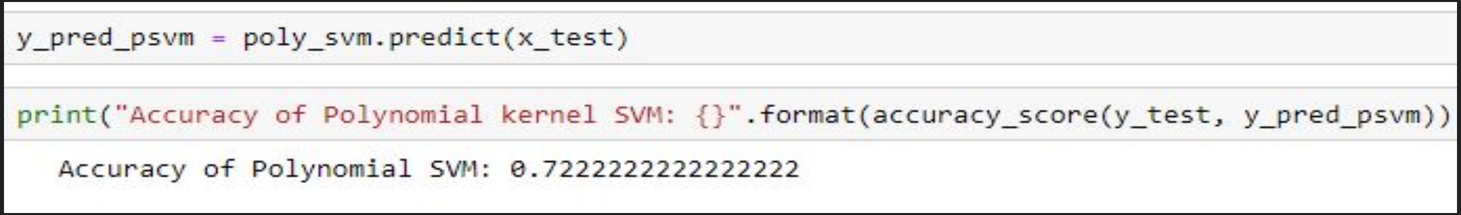
```
y_pred_lsvm = linear_svm.predict(x_test)
```

```
print("Accuracy of Linear SVM: {}".format(accuracy_score(y_test, y_pred_lsvm)))
```

```
Accuracy of Linear SVM: 0.7777777777777778
```

SVM with Polynomial Kernel

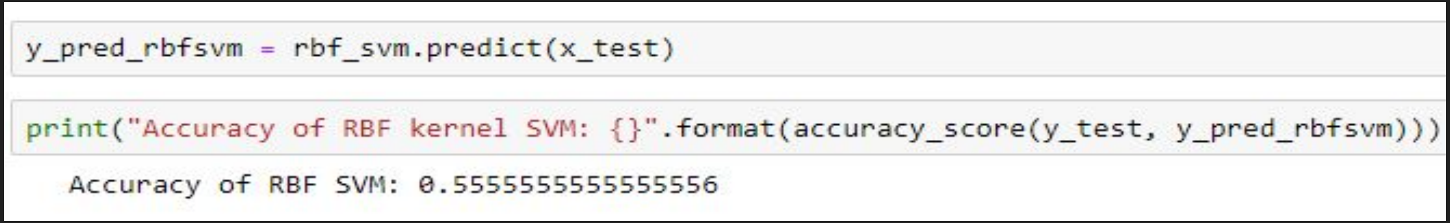
- A SVM with polynomial kernel was created using the sklearn library.
- The degree of the polynomial was chosen to be 2.
- The accuracy of the kernel SVM with polynomial kernel model developed was obtained to be 72.22%
- The accuracy remained constant for various other degree values of the polynomial kernel



```
y_pred_psvm = poly_svm.predict(x_test)

print("Accuracy of Polynomial kernel SVM: {}".format(accuracy_score(y_test, y_pred_psvm))

   Accuracy of Polynomial SVM: 0.7222222222222222
```

SVM with RBF kernel

- A SVM with RBF kernel was created using the sklearn library.
- The gamma hyper-parameter of the rbf kernel was set to be 0.1
- The accuracy of the kernel SVM with rbf kernel model developed was obtained to be 55.55%
- The accuracy remained constant for various other values of gamma



```
y_pred_rbfsvm = rbf_svm.predict(x_test)
```

```
print("Accuracy of RBF kernel SVM: {}".format(accuracy_score(y_test, y_pred_rbfsvm)))
```

```
    Accuracy of RBF SVM: 0.5555555555555556
```

Combining trivial models via ensembling.

- The highest accuracy obtained by training the trivial models was 77.777%.
- It was obtained by training the KNN model with k = 3 and Linear SVM model.
- Now, we move on to our aim 3 and 4 of increasing the accuracy of prediction.
- We do so, by combining various trivial models learnt.
- For now,

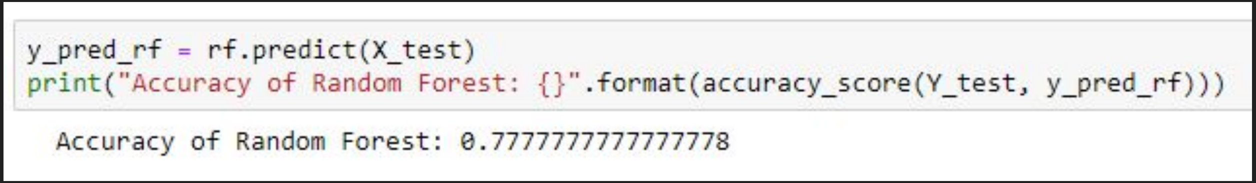
SVM model = KNN model > Naive Bayesian model > Decision Tree Algorithm.

Ensembling

- Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.
- Bagging: Bootstrapping + Aggregating. It combines bootstrapping and aggregation to form one ensemble model.
- Random Forest: they decide to split based on random selection of features.
- Voting Classifier: It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.
- AdaBoost Classifier: meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases
- Stacking Classifier: uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms.

Random Forest Classifier

- A Random Forest Classifier was created using the sklearn library, with the number of estimators as 5 and the criterion for splitting as ‘gini index’
- The model was trained on X_train and Y_train and then prediction was made using the X_test and Y_test part.
- The accuracy of the developed model was obtained to be 77.777%
- The accuracy obtained was about 8% higher than just using the trivial decision tree algorithm.

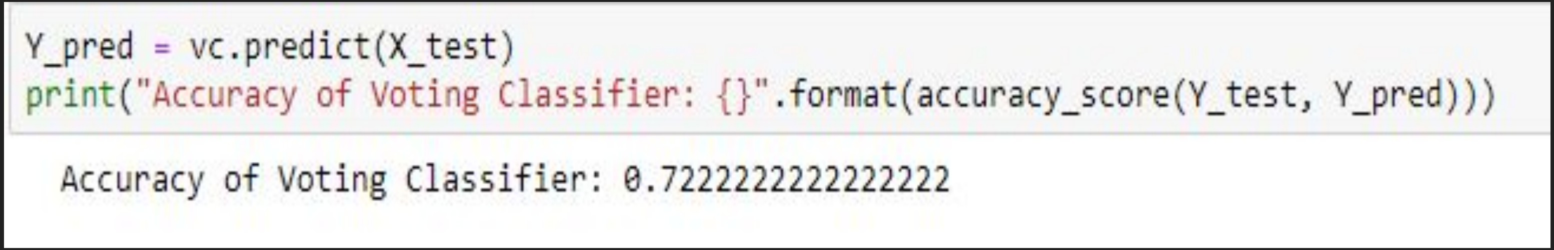


```
y_pred_rf = rf.predict(X_test)
print("Accuracy of Random Forest: {}".format(accuracy_score(Y_test, y_pred_rf)))
```

```
Accuracy of Random Forest: 0.7777777777777778
```

Voting Classifier

- A voting classifier was developed using the sklearn library by combining the models of K-nearest neighbors, Naive Bayesian and decision tree algorithm.
- The model was trained on X_train and Y_train and tested on X_test and Y_test respectively.
- The accuracy of the developed voting classifier was found to be 72.22% which is same as the random forest generator.

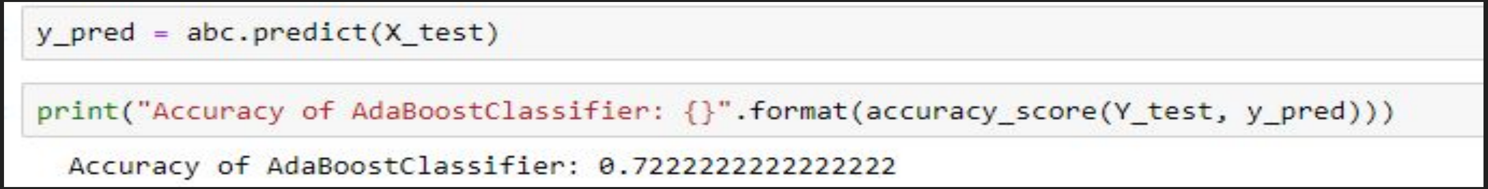


```
Y_pred = vc.predict(X_test)
print("Accuracy of Voting Classifier: {}".format(accuracy_score(Y_test, Y_pred)))
```

```
Accuracy of Voting Classifier: 0.7222222222222222
```

AdaBoost Classifier

- The AdaBoostClassifier was developed using the sklearn library with DecisionTreeClassifier as the estimator, number of estimators as 3 and learning rate as 0.2.
- The model was trained on X_train and Y_train and tested on X_test and Y_test.
- The accuracy of the developed model was obtained to be 72.22% which is same as the Random Forest Classifier.



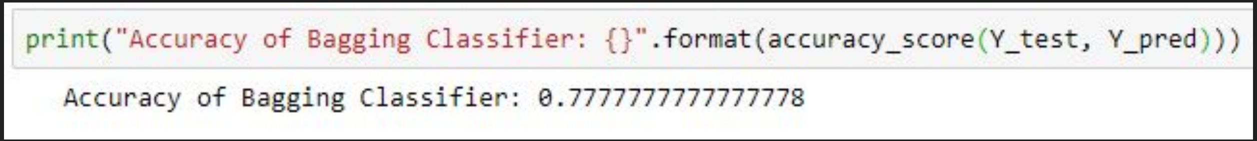
```
y_pred = abc.predict(X_test)
```

```
print("Accuracy of AdaBoostClassifier: {}".format(accuracy_score(Y_test, y_pred)))
```

```
  Accuracy of AdaBoostClassifier: 0.7222222222222222
```

Bagging Classifier

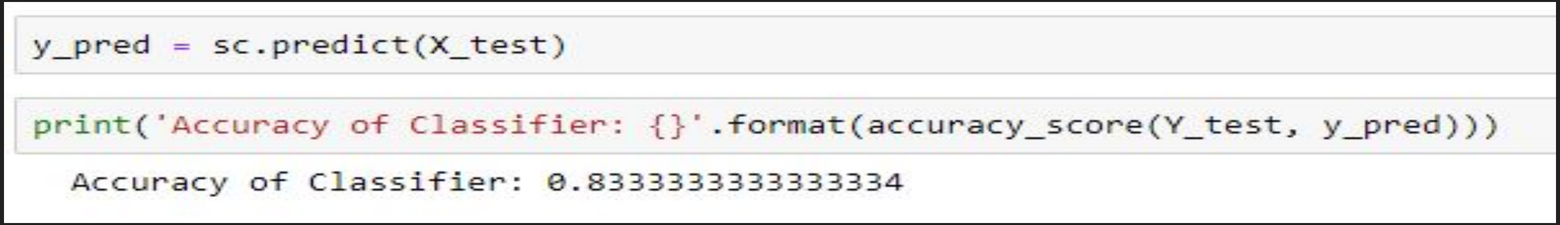
- A Bagging Classifier was created using the sklearn library, with the number of estimators as 8, max_samples = 0.5 and max_features = 1.
- The model was trained on X_train and Y_train and then prediction was made using the X_test and Y_test part.
- The accuracy of the developed model was obtained to be 77.777%
- The accuracy obtained was same as the random forest classifier.



```
print("Accuracy of Bagging Classifier: {}".format(accuracy_score(Y_test, Y_pred)))

    Accuracy of Bagging Classifier: 0.7777777777777778
```

Stacking classifier

- The stacking classifier was developed using the sklearn library by stacking the estimators of decision tree, naive bayesian and k-nearest neighbors and the final estimator was considered to be naive bayesian.
- The model was trained on X_train and Y_train and tested on X_test and Y_test.
- The accuracy of the developed model was obtained to be 83.33%



```
y_pred = sc.predict(X_test)
```

```
print('Accuracy of Classifier: {}'.format(accuracy_score(Y_test, y_pred)))
```

```
Accuracy of Classifier: 0.8333333333333334
```

Conclusions

- Upon combining the models to generate higher accuracy of prediction only one models was successful, namely:
  - Stacking Classifier

- Stacking classifier yielded an accuracy of 83.33% about 6% higher than the KNN's accuracy whose threshold we wanted to cross.