ELG 5142 Ubiquitous Sensing and Smart City

Assignment Two
2022 Summer
Group-18

Code LINK: [smart-cities-assignment2]

---

1. Use the provided dataset with fake tasks and legitimate tasks for this assignment

```
#read dataset from csv

dataset = pd.read_csv('MCSDatasetNEXTCONLab.csv')
dataset.head(10)
```

Show the first 10 rows from the dataset

	ID	Latitude	Longitude	Day	Hour	Minute	Duration	RemainingTime	Resources	Coverage	OnPeakHours	GridNumber	Ligitimacy
0	1	45.442142	-75.303369	1	4	13	40	40	9	91	0	131380	1
1	1	45.442154	-75.304366	1	4	23	40	30	9	91	0	131380	1
2	1	45.442104	-75.303963	1	4	33	40	20	9	91	0	121996	1
3	1	45.441868	-75.303577	1	4	43	40	10	9	91	0	121996	1
4	2	45.447727	-75.147722	2	15	49	30	30	5	47	0	140784	1
5	2	45.447747	-75.147951	2	15	59	30	20	5	47	0	140784	1
6	2	45.447790	-75.148303	2	16	9	30	10	5	47	0	140784	1
7	3	45.508896	-75.259807	2	12	27	30	30	4	43	0	243994	1
8	3	45.508748	-75.260652	2	12	37	30	20	4	43	0	243994	1
9	3	45.508082	-75.260380	2	12	47	30	10	4	43	0	243994	1

Show the count of each target labels

```
# showing the number of rows of target [0 or 1]
dataset["Ligitimacy"].value_counts()
```

```
1    12587
0     1897
Name: Ligitimacy, dtype: int64
```

The data unbalanced as the above figure shows, so we performed over sampling technique to balance the data.

```
#split the dataset into independent variables and target
X=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1]
```

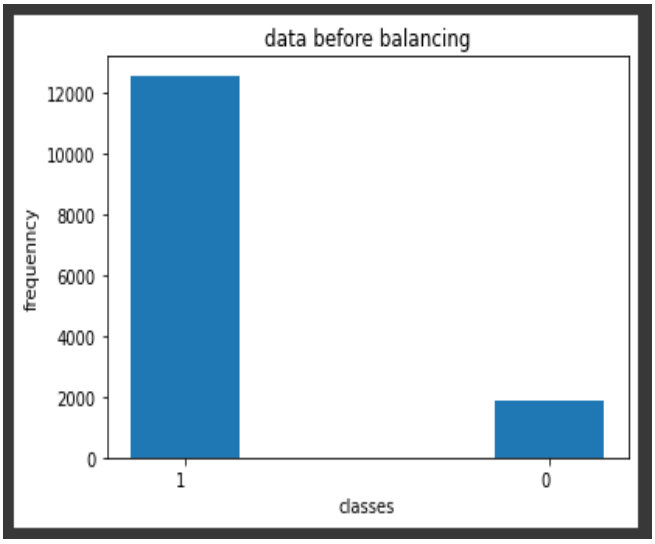
do oversampling method to make targets counts equal

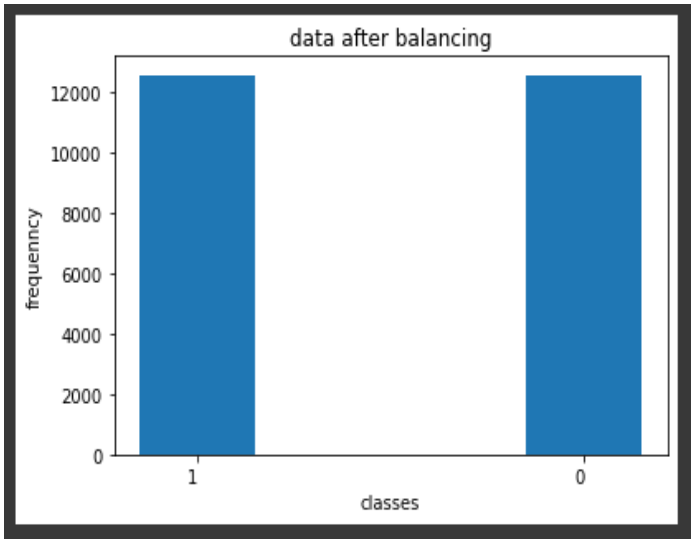
```
# define oversampling strategy
oversample = RandomOverSampler(sampling_strategy='minority')
# fit and apply the transform
X_over, y_over = oversample.fit_resample(X, y)
```

```
#split the dataset into independent variables and target
X=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1]

# define oversampling strategy
oversample = RandomOverSampler(sampling_strategy='minority')
# fit and apply the transform
X_over, y_over = oversample.fit_resample(X, y)
```

Plot the dataset before and after oversampling

```
def show_data_balanceing(title, lst_names, lst_cls ):
  plt.bar(lst_names,lst_cls,width=0.3)
  plt.title(title)
  plt.xlabel('classes')
  plt.xticks(ha='right')
  plt.ylabel('frequenncy')
  plt.show()
```





2. Split the dataset into two for training (80%) and testing (20%)

```
#split dataset into trianing and testing with 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split( X_over, y_over, test_size=0.2, random_state=42)
```

3. Train individual MLs separately using training dataset.

4. Apply test dataset to get prediction results by trained MLs (e.g., RF, Adaboost, and NB).

3.1 Random Forest Classifier

```
#Random Forest Classifier
#We got best performance at max_depth= 23
RFclf = RandomForestClassifier(max_depth=23, random_state=0)
RFclf.fit(X_train, y_train)

# Testing accuracy of RF
ypredRF_testing =RFclf.predict(X_test)

# Training accuracy of RF is X
ypredRF_training=RFclf.predict(X_train)

#plot confusion matrix
getConfusionMatrix(RFclf,X_test,y_test,"Confusion Matrix for {}".format("Random Forest Model"))

#Classification Report for Random Forest Model
print('\nClassification Report for Random Forest Model')
print(classification_report(y_test, ypredRF_testing,  target_names = ['class 0','leg class 1']))
```

```
Classification Report for Random Forest Model
              precision    recall  f1-score   support

     class 0       1.00      1.00      1.00      2505
 leg class 1       1.00      1.00      1.00      2530

    accuracy                           1.00      5035
   macro avg       1.00      1.00      1.00      5035
weighted avg       1.00      1.00      1.00      5035
```

```
# Training accuracy of Random Forest
X = accuracy_score(y_train , ypredRF_training)
print(f'Training accuracy of Random Forest: {X}')
```

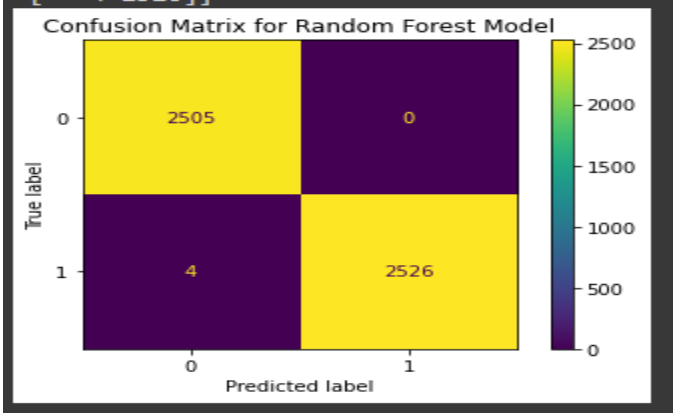
```
Training accuracy of Random Forest: 1.0
```

```
# testing accuracy of Random Forest
RandomForest_test_acc = accuracy_score(y_test, ypredRF_testing)
print(f'Testing accuracy of Random Forest: {RandomForest_test_acc}')
```

```
Testing accuracy of Random Forest: 0.9992055610724926
```

Confusion matrix for random forest testing model

```
[[2505    0]
 [   4 2526]]
```



3.2 AdaBoost Classifier

```
#AdaBoost Classifier
#We got best performance at n_estimators=2380
Adaclf = AdaBoostClassifier(n_estimators=2380, random_state=0)
Adaclf.fit(X_train, y_train)
#Testing accuracy of Adaboost
ypredAda_testing=Adaclf.predict(X_test)
#Training accuracy of Adaboost is Y
ypredAda_training=Adaclf.predict(X_train)
#plot confusion matrix
getConfusionMatrix(Adaclf,X_test,y_test,"Confusion Matrix for {}".format("AdaBoost Model"))
#Classification Report for AdaBoost Model
print('\nClassification Report for AdaBoost Model')
print(classification_report(y_test, ypredAda_testing,  target_names = ['class 0','leg class 1']))
```

```
Classification Report for AdaBoost Model
              precision    recall  f1-score   support

     class 0       1.00      1.00      1.00      2505
 leg class 1       1.00      1.00      1.00      2530

    accuracy                           1.00      5035
   macro avg       1.00      1.00      1.00      5035
weighted avg       1.00      1.00      1.00      5035
```

```
# Training accuracy of Adaboost
Y = accuracy_score(y_train, ypredAda_training)
print(f'Training accuracy of Adaboost: {Y}')
```

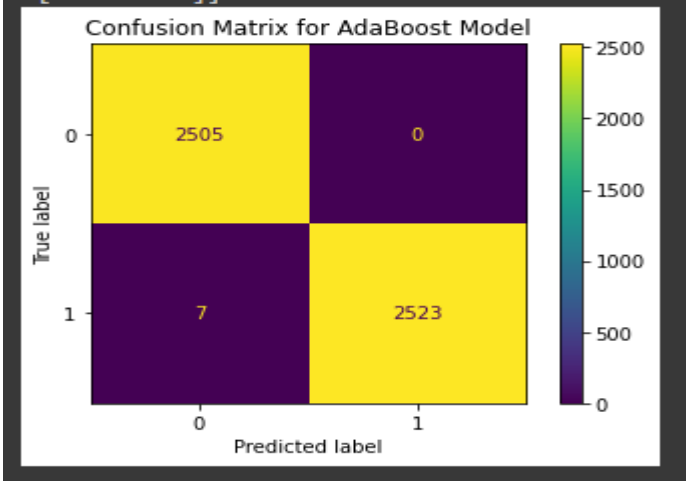
```
Training accuracy of Adaboost: 1.0
```

```
# testing accuracy of Adaboost
adaBoost_test_acc = accuracy_score(y_test, ypredAda_testing)
print(f'Testing accuracy of Adaboost: {adaBoost_test_acc}')
```

```
Testing accuracy of Adaboost: 0.998609731876862
```

Confusion matrix for adaboost testing model

```
[[2505    0]
 [   7 2523]]
```



3.3 Naive Bayes Bernoulli Classifier

Note that we used Naïve Bayes Bernoulli not Gaussian because:

Bernoulli Naive Bayes is good at handling Boolean/binary attributes, while Multinomial Naive Bayes is good at handling discrete values and Gaussian naive Bayes is good at handling continuous values.

```
#Naive Bayes Bernoulli Classifier
#we used Bernoulli here because our problem target is 0 or 1 (binary classification)
#we also used Gaussian Naive Bayes classifier and we got the same accuracy
Bernoulliclf = BernoulliNB()
Bernoulliclf.fit(X_train, y_train)
#Training accuracy of Naïve Bayes
ypredBernoulli_testing=Bernoulliclf.predict(X_test)
#Training accuracy of Naïve Bayes  is Z
ypredBernoulli_training=Bernoulliclf.predict(X_train)

#plot confusion matrix
getConfusionMatrix(Bernoulliclf,X_test,y_test,"Confusion Matrix for {}".format("Bernoulli Naive Bayes Model"))
#Classification Report for Bernoulli Naive Bayes Model
print('\nClassification Report for Bernoulli Naive Bayes Model')
print(classification_report(y_test, ypredBernoulli_testing,  target_names=  ['class 0','leg class 1']))
```

```
Classification Report for Bernoulli Naive Bayes Model
              precision    recall  f1-score   support

     class 0       0.77      0.48      0.59      2505
 leg class 1       0.62      0.86      0.72      2530

    accuracy                           0.67      5035
   macro avg       0.70      0.67      0.66      5035
weighted avg       0.70      0.67      0.66      5035
```

```
# Training accuracy of Naive Bayes Bernoulli Classifier
Z = accuracy_score(y_train, ypredBernoulli_training)
print(f'Training accuracy of Naive Bayes Bernoulli Classifier: {Z}')
```

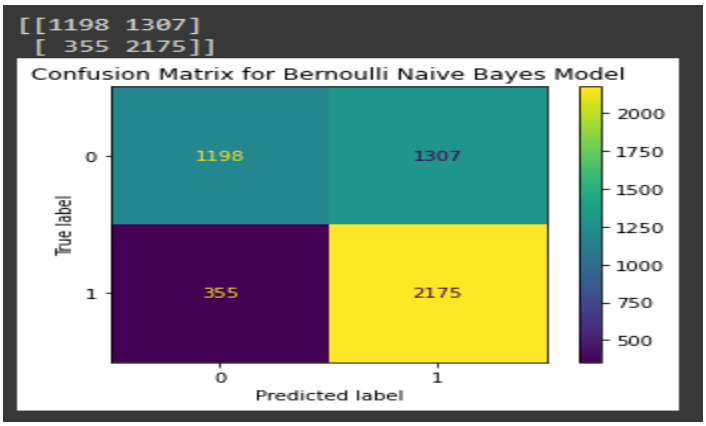
```
Training accuracy of Naive Bayes Bernoulli Classifier: 0.6759521326778887
```

```
# testing accuracy of Naive Bayes Bernoulli
NaiveBayes_test_acc = accuracy_score(y_test, ypredBernoulli_testing)
print(f'Testing accuracy of Naive Bayes Bernoulli: {NaiveBayes_test_acc}')
```

```
Testing accuracy of Naive Bayes Bernoulli: 0.6699106256206554
```

Confusion matrix for Naïve-Bayes Bernoulli testing model

```
[[1198 1307]
 [ 355 2175]]
```



5. Use the majority to make final decision for each task

```
# ypredRF_training, ypredAda, ypredBernoulli
lstvoct = []
AggOutputVotinglst = []
for i in range(0,len(ypredAda_testing)):
  sum = ypredRF_testing[i]+ ypredAda_testing [i] + ypredBernoulli_testing[i]
  if (sum <= 1):
    lstvoct.append(0)
  else:
    lstvoct.append(1)
  AggOutputVotinglst.append(sum)
```

	Prediction of RF	Prediction of Adaboost	Prediction of NB	Aggregated output of voting	Voting Result
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	1	1	0
3	1	1	1	3	1
4	1	1	1	3	1
...	...	...	...	...	...
5030	0	0	1	1	0
5031	0	0	1	1	0
5032	0	0	0	0	0
5033	1	1	1	3	1
5034	1	1	1	3	1

5035 rows × 5 columns

```
Classification Report for Ensemble-vote
              precision    recall  f1-score   support

     class 0       1.00      1.00      1.00      2505
 leg class 1       1.00      1.00      1.00      2530

    accuracy                           1.00      5035
   macro avg       1.00      1.00      1.00      5035
weighted avg       1.00      1.00      1.00      5035
```

```
Testing accuracy of Ensemble-vote: 0.9992055610724926
```

We have calculated the summation of the prediction so we can get the voting result and then build our condition if the summation of the votes were less than or equal one then we assigned the result with zero value, and one otherwise.

6. Use the weighted sum aggregation to make final decision for each task

```
Total = X + Y + Z
# The Weight of random forcest (wRF = X/(X+Y+Z))
wRF = X / Total
print(f'The Weight of random forcest(wRF)       : {wRF}')
# The Weight of Adaboost (wAdaboost = Y/(X+Y+Z))
wAdaboost = Y / Total
print(f'The Weight of Adaboost(wAdaboost)       : {wAdaboost}')
# The Weight of Naive Bayes Bernoulli (wNB = Z/(X+Y+Z))
wNB = Z / Total
print(f'The Weight of Naive Bayes Bernoulli(wNB): {wNB}')
```

```
The Weight of random forcest(wRF)       : 0.37369876231652777
The Weight of Adaboost(wAdaboost)       : 0.37369876231652777
The Weight of Naive Bayes Bernoulli(wNB): 0.25260247536694436
```

```
# ypredRF_training, ypredAda, ypredBernoulli
FinalDecision = []
AggOutputlst = []
for i in range(0, len(ypredRF_testing)):
  AggOutput = ypredRF_testing[i] * wRF + ypredAda_testing[i] * wAdaboost + ypredBernoulli_testing[i] * wNB
  AggOutputlst.append(AggOutput)
  if(AggOutput > 0.5):
    FinalDecision.append(1)
  else:
    FinalDecision.append(0)
```

While building the three models, we calculated the accuracy of the training as required in the assignment and then we built the equations, just to reach the aggregated output and finally we got the result regarding to the aggregated output.

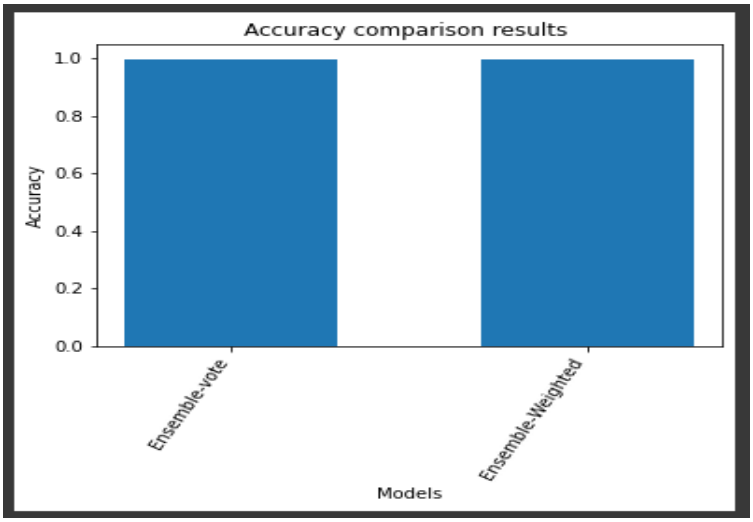
	Prediction of RF	Prediction of Adaboost	Prediction of NB	Aggregated output	*Final Decision*
0	0	0	0	0.000000	0
1	0	0	0	0.000000	0
2	0	0	1	0.252602	0
3	1	1	1	1.000000	1
4	1	1	1	1.000000	1
...	...	...	...	...	...
5030	0	0	1	0.252602	0
5031	0	0	1	0.252602	0
5032	0	0	0	0.000000	0
5033	1	1	1	1.000000	1
5034	1	1	1	1.000000	1

5035 rows × 5 columns

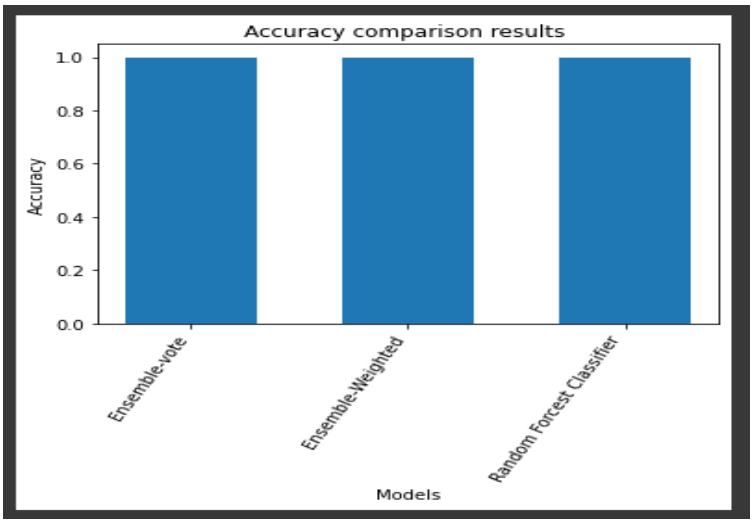
```
Classification Report for Ensemble-Weighted
              precision    recall  f1-score   support

     class 0       1.00      1.00      1.00      2505
 leg class 1       1.00      1.00      1.00      2530

    accuracy                           1.00      5035
   macro avg       1.00      1.00      1.00      5035
weighted avg       1.00      1.00      1.00      5035
```

```
Testing accuracy of Ensemble-Weighted: 0.9992055610724926
```

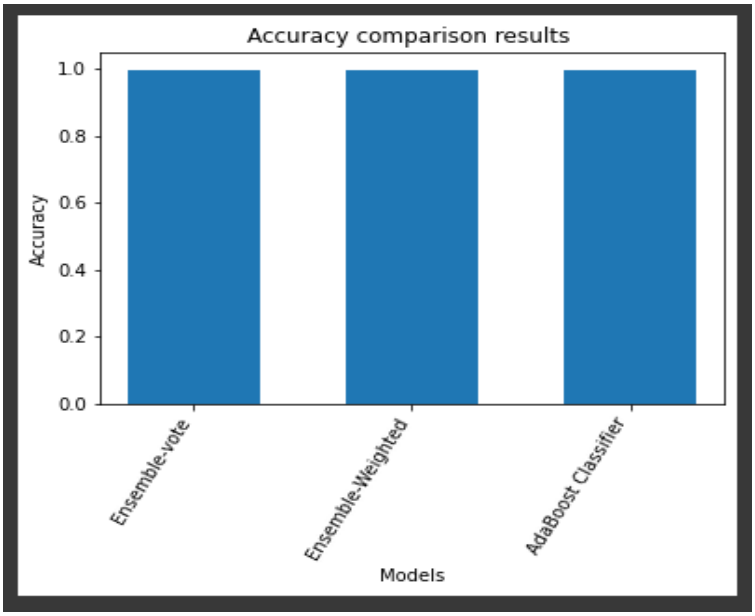
7. Compare the two-ensemble framework performance to each other, and with individual ML models.



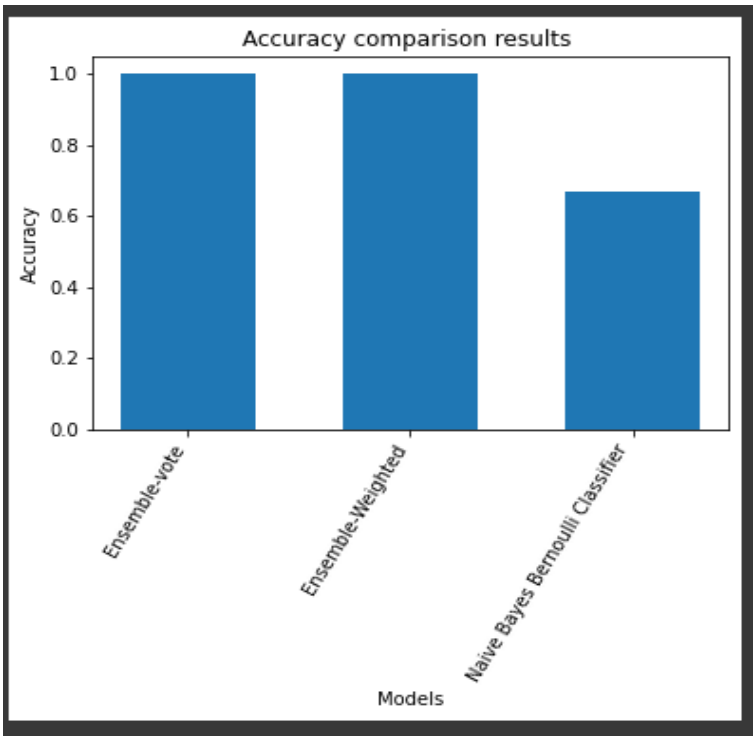
	Testing accuracy
Ensemble-vote	0.999206
Ensemble-Weighted	0.999206



	Testing accuracy
Ensemble-vote	0.999206
Ensemble-Weighted	0.999206
Random Forcest Classifier	0.999206

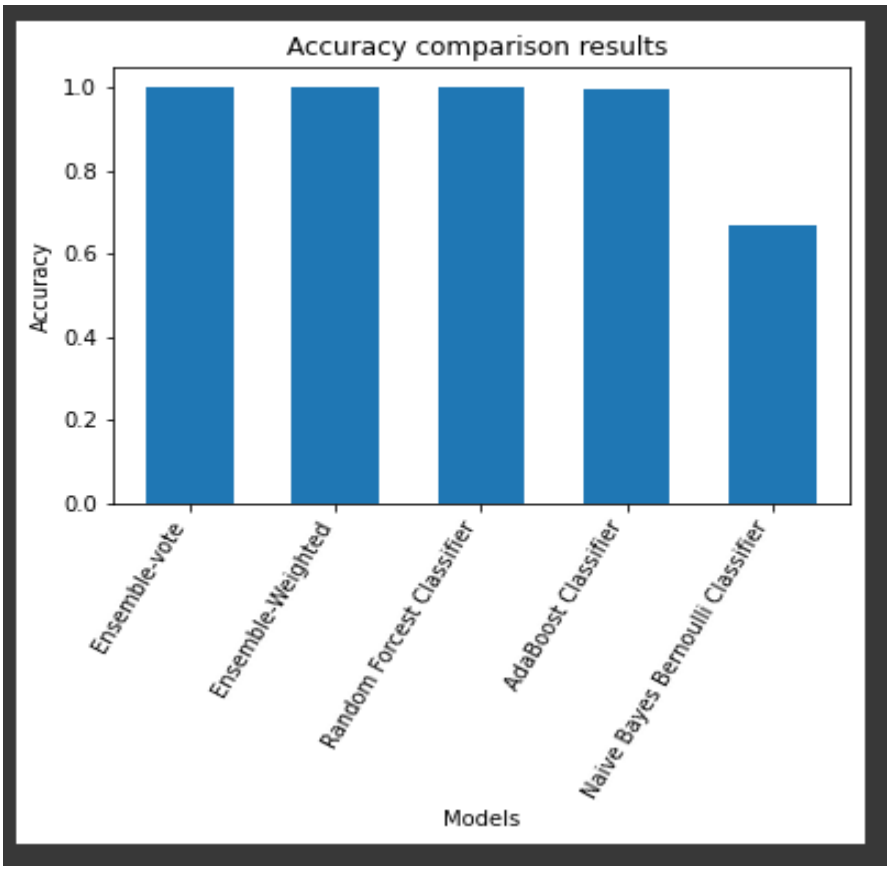


	Testing accuracy
Ensemble-vote	0.999206
Ensemble-Weighted	0.999206
AdaBoost Classifier	0.998610



	Testing accuracy
Ensemble-vote	0.999206
Ensemble-Weighted	0.999206
Naive Bayes Bernoulli Classifier	0.669911

8. Plot a bar chart figure to show accuracy comparison results.



	Testing accuracy
Ensemble-vote	0.999206
Ensemble-Weighted	0.999206
Random Forcest Classifier	0.999206
AdaBoost Classifier	0.998610
Naive Bayes Bernoulli Classifier	0.669911

Conclusion

we got a higher models' accuracies while using ensemble-voting mechanism, ensemble-weighted mechanism, Random-forest classifier and Adaboost classifier, but we got a bad accuracy while using Naïve-Bayes Bernoulli classifier.

However, the implementation is different, the results of ensemble-voting mechanism and ensemble-weighted mechanism are the same.