Karthik Bangalore Mani
A20344597

Phase III Report

- **Choose_the_best_classifer() :**
  I have chosen the best classifier based on the below hypothesis:

  Among multiple classifiers, a classifier is said to be the best, if, the sum of probabilities of its mis-predictions is the minimum among all the other classifiers.

  I have followed the below 4 steps to figure out the best classifier:

  1. **Train 3 different classifiers on the training dataset <X_train, y_train> :**

```
bern_clf = BernoulliNB()
bern_clf.fit(X_train, y_train)

logi_clf = LogisticRegression()
logi_clf.fit(X_train, y_train)

svc_clf = SVC(degree=4,probability=True,random_state=0)
svc_clf.fit(X_train, y_train)
```

  2. **Append the 3 trained classifiers into a list called *classifersList* :**

```
classifersList.append(bern_clf)
classifersList.append(logi_clf)
classifersList.append(svc_clf)
```

  3. **Iterate over the classifersList and find out which classifier mis-predicts the least.**
     The classifier with minimum sum of probabilities of mis-predictions is chosen as the best classifer. This type of classifier is identified with the help of getWrongPredProbSum() .

```
for classifer in classifersList:
    X = classifer.predict(X_val)
    Xprob = classifer.predict_proba(X_val)
    wrong_pred_prob_sum = x.getWrongPredProbSum(X,y_val,Xprob)

    if (wrong_pred_prob_sum) < best_classifier_wrong_prob_sum:
        best_classifier_wrong_prob_sum = wrong_pred_prob_sum
        bestClassifier = classifer
```

  4. **Return a new Un-trained object of the best classifer :**

```
if bestClassifier == bern_clf :
    return BernoulliNB()
elif bestClassifier == logi_clf :
    return LogisticRegression()
else :
    return SVC(degree=4,probability=True,random_state=0)
```

- **getWrongPredProbSum():**
  This method return the sum of probabilities of predictions that not only just wrong, but are **confidently wrong**.

  A prediction is said to be **wrong**, if any of the below 2 criteria are satisfied:
  1. If the actual class is Excellent and the predicted class is Trash
  2. If the actual class is Trash and the predicted class is Excellent

  In other words, if the predicted class is not equal to the actual class, then the prediction is wrong.

  A prediction is said to be **confidently wrong**, if it satisfies both of the below 2 criteria:
  1. Prediction is wrong
  2. Probability of the wrong prediction is greater than 60%.

  Note: I decided on fixing the confidence level at 60% after several experiments of trial and error.

  Steps to sum of probabilities of the wrong predictions which have a confidence level more than 60% :

  1. **Iterate over all the predictions.**
  2. **During the iteration, if a wrong prediction is encountered, the probability of wrong prediction is more than 60%, then add the probability of wrong prediction (predicted_proba) to wrong_pred_prob_sum.**
  3. **Return wrong_pred_prob_sum.**

  Code Snippet:

```
def getWrongPredProbSum(self,predicted,actual,predicted_proba):
    wrong_pred_prob_sum = 0
    #Step 1 : Iterate over all predictions.
    for i in range(0,len(predicted)):
        #Step 2 : Check if it is a wrong prediction.
        if (predicted[i] != actual[i]):
            #If actual class is Trash, then wrong prediction is Excellent.
            #Therefore, consider the probability of Excellent, which is  predicted_proba[i][0] .
            if (actual[i] == 'Trash'):
                if (predicted_proba[i][0] > 0.6):
                    wrong_pred_prob_sum = wrong_pred_prob_sum + predicted_proba[i][0]
            #If actual class is Excellent, then wrong prediction is Trash.
            #Therefore, consider consider the probability of Trash, which is predicted_proba[i][1] .
            elif (actual[i] == 'Excellent'):
                if (predicted_proba[i][1] > 0.6):
```

```
                wrong_pred_prob_sum = wrong_pred_prob_sum + predicted_proba[i][1]
    #Step 3 : Return wrong_pred_prob_sum
    return wrong_pred_prob_sum
```

- **Simulation Results**

SIMULATION RESULTS ON dataset1

Wealth (the larger the better)

Agent_bnb:	$1,775,950.00
Agent_lr:	$1,638,100.00
Agent_svc:	$1,680,600.00
Agent_kbangal2:	$1,775,950.00

Log-loss (the smaller the better)

Agent_bnb:	332.25
Agent_lr:	373.89
Agent_svc:	364.92
Agent_kbangal2:	332.25

0/1 Loss (the smaller the better)

Agent_bnb:	88
Agent_lr:	115
Agent_svc:	112
Agent_kbangal2:	88

SIMULATION RESULTS ON dataset2

Wealth (the larger the better)

Agent_bnb:	$1,507,950.00
Agent_lr:	$1,717,100.00
Agent_svc:	$1,631,600.00
Agent_kbangal2:	$1,717,100.00

Log-loss (the smaller the better)

Agent_bnb:	553.13
Agent_lr:	487.71
Agent_svc:	507.76
Agent_kbangal2:	487.71

Karthik Bangalore Mani
A20344597

0/1 Loss (the smaller the better)

Agent_bnb:	250
Agent_lr:	223
Agent_svc:	236
Agent_kbangal2:	223

SIMULATION RESULTS ON dataset3

Wealth (the larger the better)

Agent_bnb:	$795,950.00
Agent_lr:	$810,100.00
Agent_svc:	$819,600.00
Agent_kbangal2:	$819,600.00

Log-loss (the smaller the better)

Agent_bnb:	571.94
Agent_lr:	566.91
Agent_svc:	560.14
Agent_kbangal2:	560.14

0/1 Loss (the smaller the better)

Agent_bnb:	250
Agent_lr:	255
Agent_svc:	252
Agent_kbangal2:	252