YAHOO TROLL QUESTION DETECTION

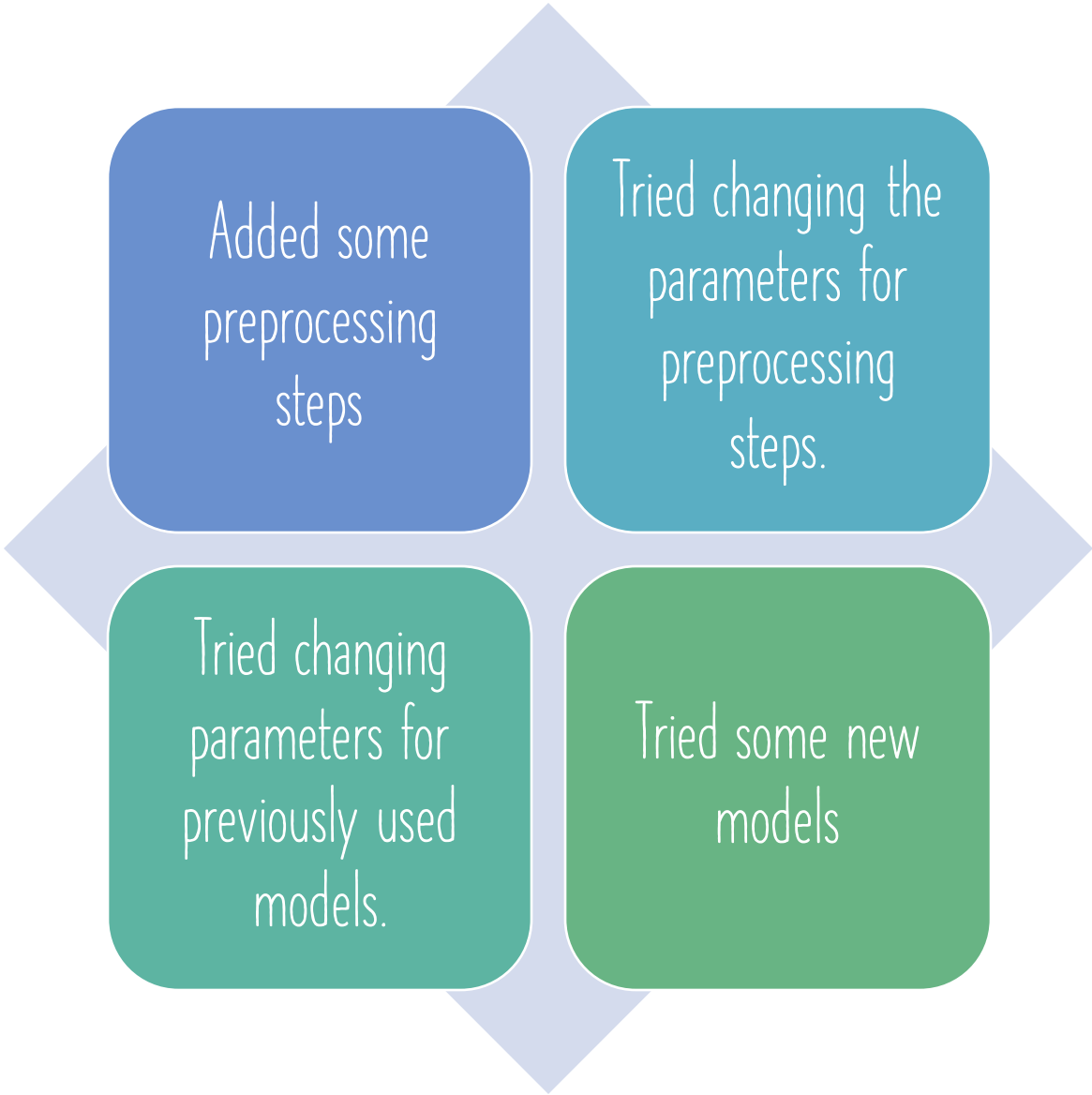
Karanjit Saha (IMT2020003)

Netradeepak Chinchwadkar (IMT2020014)

WHAT WE DID BEFORE

Just a recap of what we had done before the previous evaluation

WHAT WE DID AFTER FIRST SUBMISSION



THE NEW STUFF

ADDITIONS IN PREPROCESSING STEPS:-

We added spell checker.

We removed Non-english words.

- TRIED CHANGING THE PARAMETERS FOR PREPROCESSING STEPS.

Tried using tfidfVectoriser. The f1 score dropped and the kaggle score also dropped. Hence, we are sticking to CountVectoriser.

We are now not converting text to lowercase.

Tried doing char tokenization(by using analyzer attribute) didn't improve the f1 score so dropped the idea.

Tried using HashingVectoriser by sklearn. Didn't increase f1 score so dropped the idea.

- TRIED CHANGING THE PARAMETERS FOR PREPROCESSING STEPS.

Tried different values for n-grams parameter of CountVectorise, like (1,4), but then the model was overfitting.

Tried limiting the number of features in CountVectorizer. Didn't improve the f1 score much so dropped the idea.

Tried using max_df and min_df attributes of CountVectoriser, but then again it decreased the f1 score of the model. Hence, we dropped the idea.

TRIED CHANGING PARAMETERS FOR PREVIOUSLY USED MODELS.

- Used class_weights = 'balanced' in Logistic Regression.
- Used class_weights and sample_weights for other models as well.

TRIED SOME NEW MODELS

We tried using KNN model. But it took 6 hours and was still running hence we dropped it.

We tried using ADABoost model, but this again took too much of time to give the output.

We tried using Perceptron model. It gave the output but it was not better than that of Logistic Regression.

We tried using SVM . Only LinearSVC gave output in reasonable time the SVC and NuSVC took unreasonable amount of time and still didn't gave any output.

We tried XGBoost model as well. It didn't work as well as Logistic for our dataset.

```
train f1 score:  0.9991510177679853
test f1 score:  0.5642463501885333
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    750504
           1       1.00      1.00      1.00     49496

    accuracy                           1.00    800000
   macro avg       1.00      1.00      1.00    800000
weighted avg       1.00      1.00      1.00    800000
```

SVM MODEL

XGBOOST

```
train f1 score:  0.643289221187110 3
test f1 score:  0.5519982457102133
              precision    recall  f1-score   support

           0       1.00      0.93      0.96    750504
           1       0.48      0.98      0.64     49496

    accuracy                           0.93    800000
   macro avg       0.74      0.95      0.80    800000
weighted avg       0.97      0.93      0.94    800000
```

PERCEPTRON

```
train f1 score:  0.8999726950031856
test f1 score:  0.5690621193666626
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    750504
           1       0.82      1.00      0.90     49496

    accuracy                           0.99    800000
   macro avg       0.91      0.99      0.95    800000
weighted avg       0.99      0.99      0.99    800000
```

Bagging

```
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
clf = BaggingClassifier(base_estimator=LogisticRegression(
    penalty='l2', max_iter=50000, solver='lbfgs', class_weight='balanced'), n_estimators=10, random_state=0)
clf.fit(train, y_train)
y_pred = clf.predict(train)
print("train f1 score: ",f1_score(y_train,y_pred))
print("test f1 score: ",f1_score(y_test,clf.predict(test)))
test_y_pred = clf.predict(test_df_matrix)
print(metrics.classification_report(y_train,y_pred))
```

[16] ✓ 83m 45.4s

```
train f1 score:  0.9077825516970804
test f1 score:  0.6247117754631469
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    750504
           1       0.86      0.96      0.91     49496

    accuracy                           0.99    800000
   macro avg       0.93      0.97      0.95    800000
weighted avg       0.99      0.99      0.99    800000
```

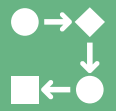
BAGGING CLASSIFIER

KEY OBSERVATIONS

The more we preprocess the data the more amount of relevant data that we lose, hence our accuracy decreases.

Hence, we tried to run our model on un-preprocessed data and got the best result uptill now.

Another observation we did was that the training dataset is biased, hence we introduced the notion of class weights and sample weights in the code.



Another observation we did was that the data is not linearly separable.

THANK YOU