Leaderboard Competition: NLP with Disaster Tweets Dataset

Project Team: HAAAT

Harish, Aryan, Ashwin, Abrielle, Tahir

December 15, 2023

CS410 Project Team *("HAAAT")*

- Tahir Bagasrawala *(tahirib2)*
- Ashwin Saxena *(ashwins2)*
- Aryan Gandhi *(aryang6)*
- Abrielle Agron *(aa106) (Captain)*
- Harish Venkata *(hkv2)*

Dataset

Dataset: High Level

Where did we get the data set from?

- Kaggle's Natural Language Processing with Disaster Tweets.

What are we predicting?

- We're predicting whether a given tweet is about a real disaster (1) or not (0).

Dataset: Details

What files are we using?

- **Training set:** `train.csv`
- **Test set:** `test.csv`
- **Sample submission file**: `sample_submission.csv`

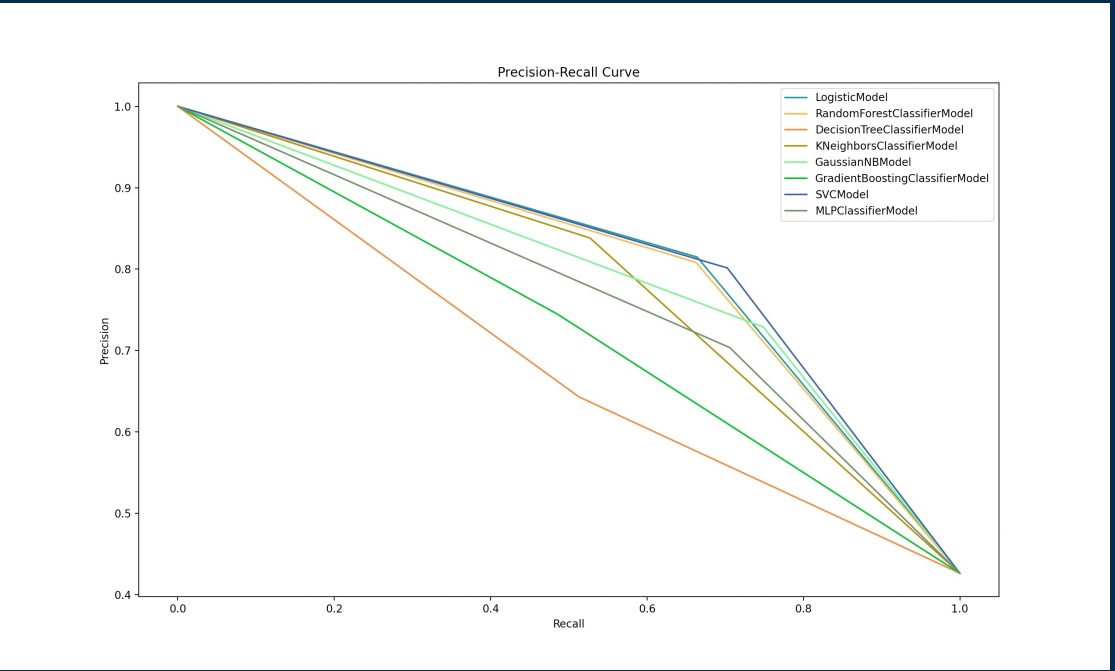
What columns does the file have?

- id - a unique identifier for each tweet
- text - the text of the tweet
- location - the location the tweet was sent from (may be blank)
- keyword - a particular keyword from the tweet (may be blank)
- target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not (0)

Purpose: Beat the Baseline Model

Leaderboard Competition Results

- Lower recall: Baseline (Logistic) is the best performer
  - SVC, Random Forest, and K-neighbors are close contenders
- Higher recall: SVC is the best performer
  - Logistic, Random Forest, and Gaussian Naïve Bayes are close contenders
- Competitor must attempt to outperform the Logistic Model on a test set



Setup Process (quick!)

Installation

1. Download or clone the following repository onto your local machine: https://github.com/tahirbags/cs410_project_public
2. Ensure you are using a version of python >= 3.6
3. Inside the repository, create a virtual environment:
   - `python[3.x.x] -m venv myvenv`
   - `source myvenv/bin/activate`
4. Inside the virtual environment, run the following:
   - `pip install scikit-learn`
   - `pip install pandas`
   - `pip install matplotlib`

Model Setup (using `sklearn`)

1. In the models/ folder, either:
    1. select an existing file (e.g. `GaussianNB.py`, `MLP.py`)
    2. create a new file named `[yourModel].py`
2. Define the class for your model, with the following three methods:
    1. `__init__(self)`
    2. `train(self, X_train, y_train)`
    3. `predict(self, X_test)`

models > GaussianNB.py > GaussianNBModel

```python
from sklearn.naive_bayes import GaussianNB

class GaussianNBModel:
    """
    A class used to represent a Gaussian Naive Bayes Classifier model

    Attributes
    ----------
    model :
        an instance of the GaussianNB Class from scikit-learn

    Methods
    -------
    train(X_train, y_train)
        Trains the model using given training data
    predict(X_test)
        Predicts labels for given test data
    """
    def __init__(self):
        """
        Hyperparameter selection:

        List of parameters that can be tuned:
        dict_keys(['priors', 'var_smoothing'])

        Best parameters found:
          {'var_smoothing': 1.0}
        """
        self.model = GaussianNB(var_smoothing=0.5)

    def train(self, X_train, y_train):
        self.model.fit(X_train.todense(), y_train)

    def predict(self, X_test):
        return self.model.predict(X_test.todense())

```

Usage

Hyperparameter Tuning on Training Set (hpo_tune.py)

1. Ensure that your model is listed in the options and if/else branches of `hpo_tune.py`.
2. In the respective branch, update the parameter space for your model.
3. Call `hpo_tune.py` and enter your model name.
4. Your model's hyperparameters will be tuned based on the parameter space you have defined, and the best settings will be printed to the console
5. Update your model with these new settings

```
elif model_to_tune == 'NaiveBayes':
    mlp_model = GaussianNB()

    parameter_space = {
    'var_smoothing': np.logspace(0,2, num=4),
    }
```

```
(myvenv) (base) ashwinsax@Ashwins-MacBook-Pro-2 cs410_project_public % python hpo_tune.py
[CAPS sensitive] Choose one model to tune:
    - MLP
    - AdaBoost
    - DecisionTree
    - NaiveBayes
    - Logistic
    - GradientBoosting
    - KNeighbors
    - SVC
    - RandomForest

Which model would like to tune?
--> NaiveBayes
List of parameters that can be trained:  dict_keys(['priors', 'var_smoothing'])
Hyperparameter tuning in progress...
Best parameters found:
 {'var_smoothing': 1.0}
Elapsed: 24.1119 seconds
```

Testing Against Other Models (model_eval.py)

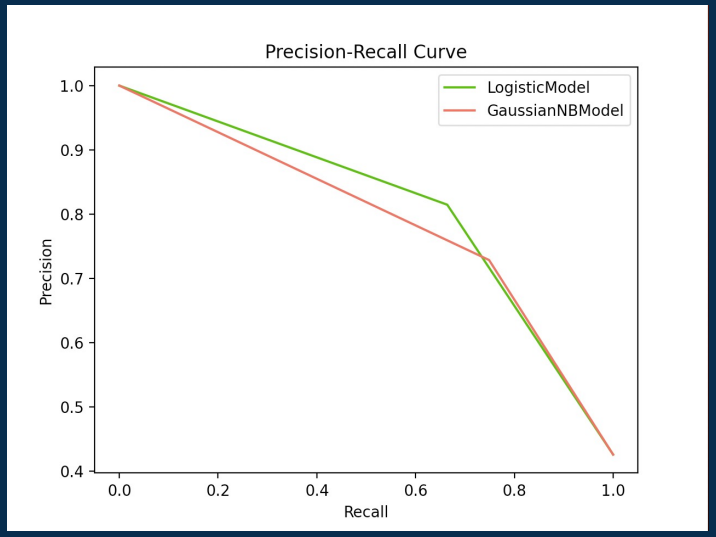
1. Ensure that your model is listed in the `models_to_evaluate` list
2. Call `model_eval.py`.
3. See your results!

```
# List of models to evaluate - each entry is a (file_path, class_name) tuple
models_to_evaluate = [
    ('models/model1.py', 'LogisticModel'),
    # ('models/RandomForestClassifier.py', 'RandomForestClassifierModel'),
    # ('models/AdaBoostClassifier.py', 'AdaBoostClassifierModel'),
    # ('models/DecisionTreeClassifier.py', 'DecisionTreeClassifierModel'),
    # ('models/KNeighborsClassifier.py', 'KNeighborsClassifierModel'), #TB

    # ('models/Kmeans.py', 'KMeansModel'),
    # ('models/LatentDirichletAllocation.py', 'LDAModel'),

    ('models/GaussianNB.py', 'GaussianNBModel'),
    # ('models/GradientBoostingClassifier.py', 'GradientBoostingClassifierModel'),
    # ('models/svc.py', 'SVCModel'),
    # ('models/mlp.py', 'MLPClassifierModel'),
]
```

```
(myvenv) (base) ashwinsax@Ashwins-MacBook-Pro-2 cs410_project_public % python model_eval.py
Results for LogisticModel: {'accuracy': 0.7925147734734077, 'precision': 0.8147448015122873, 'recall': 0.6640986132511556, 'f1_score': 0.7317
48726655348}
Elapsed: 0.1177 seconds
Results for GaussianNBModel: {'accuracy': 0.7741300065659882, 'precision': 0.7286356821589205, 'recall': 0.74884437596302, 'f1_score': 0.7386
01823708206 6}
Elapsed: 1.0938 seconds
```

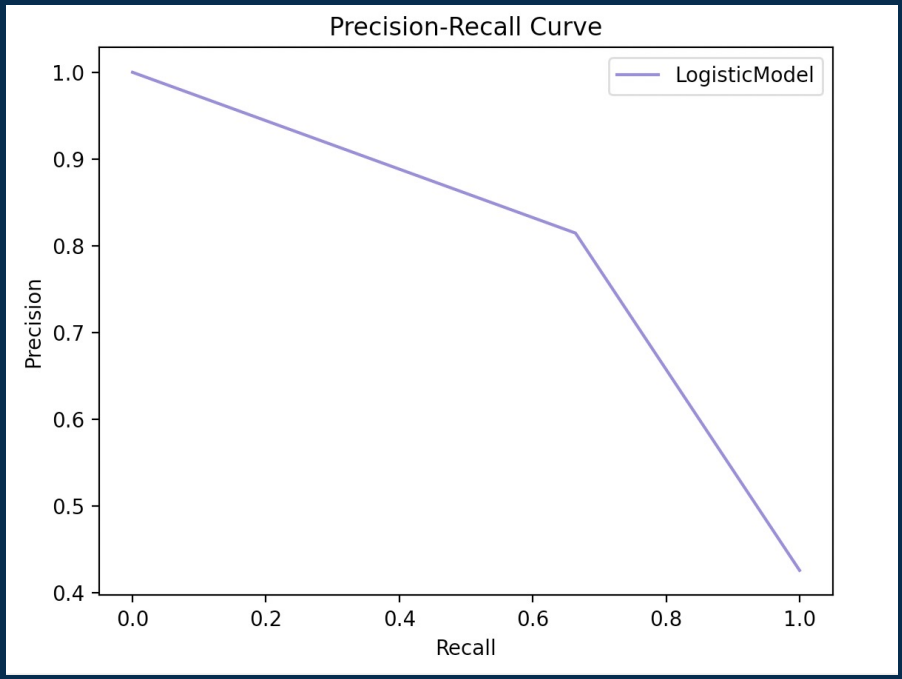


11 NLP Models Evaluated

Logistic Model - baseline

This model (also known as logit or MaxEnt) is popular for classification problems and is our baseline for this project

```
{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.7317487266553348}
Elapsed: 0.0932 seconds
```



Random Forest Classifier

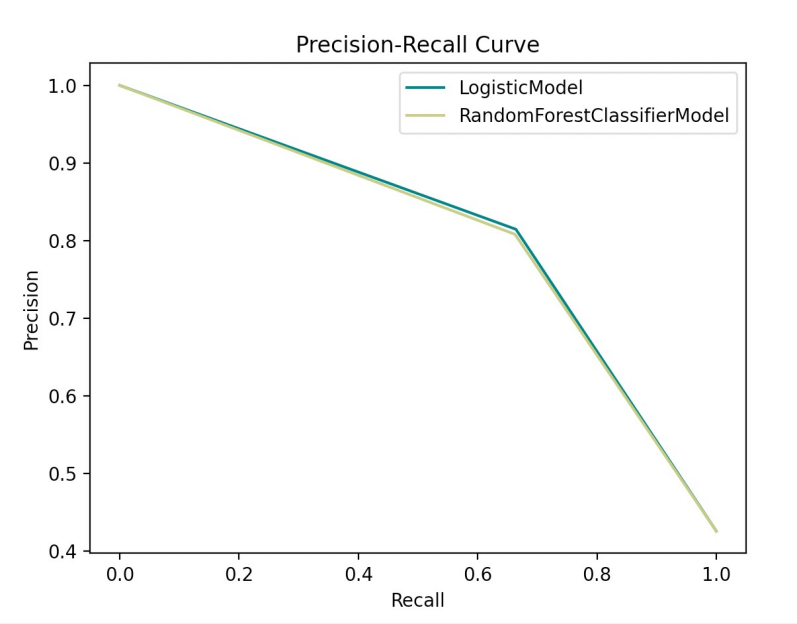
This model is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.7317487266553348}
Elapsed: 0.0932 seconds
```

```
Results for RandomForestClassifierModel:

{'accuracy': 0.7892317793827971,
'precision': 0.8082706766917294,
'recall': 0.662557781201849,
'f1_score': 0.7281964436917866}
Elapsed: 13.5767 seconds
```



AdaBoost Classifier

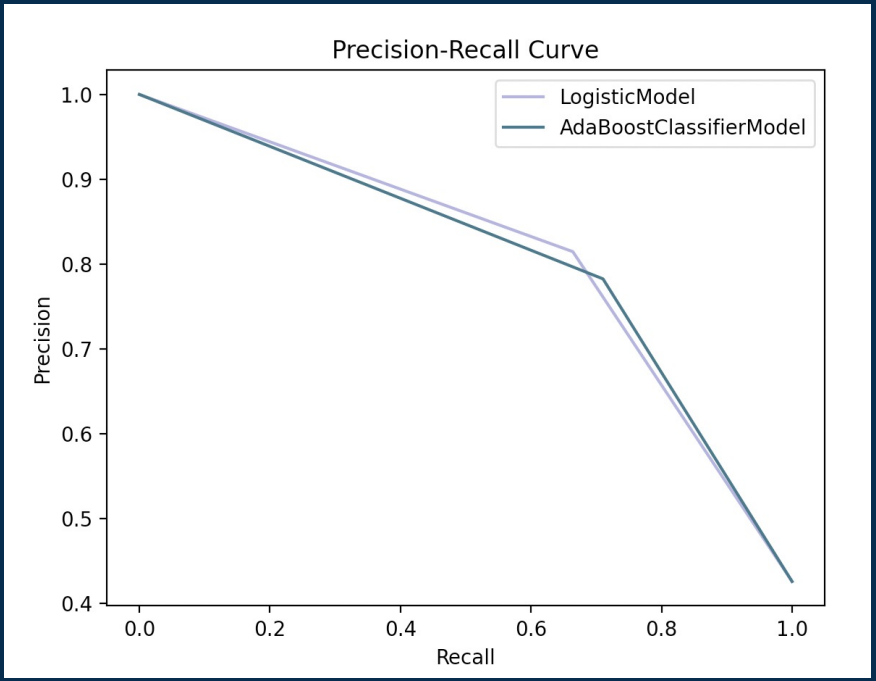
This model is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.7317487266553348}
Elapsed: 0.0932 seconds
```

```
Results for AdaBoostClassifierModel:

{'accuracy': 0.7925147734734077,
'precision': 0.7826825127334465,
'recall': 0.7103235747303543,
'f1_score': 0.7447495961227785}
Elapsed: 780.157 seconds
```



K-Neighbors Classifier

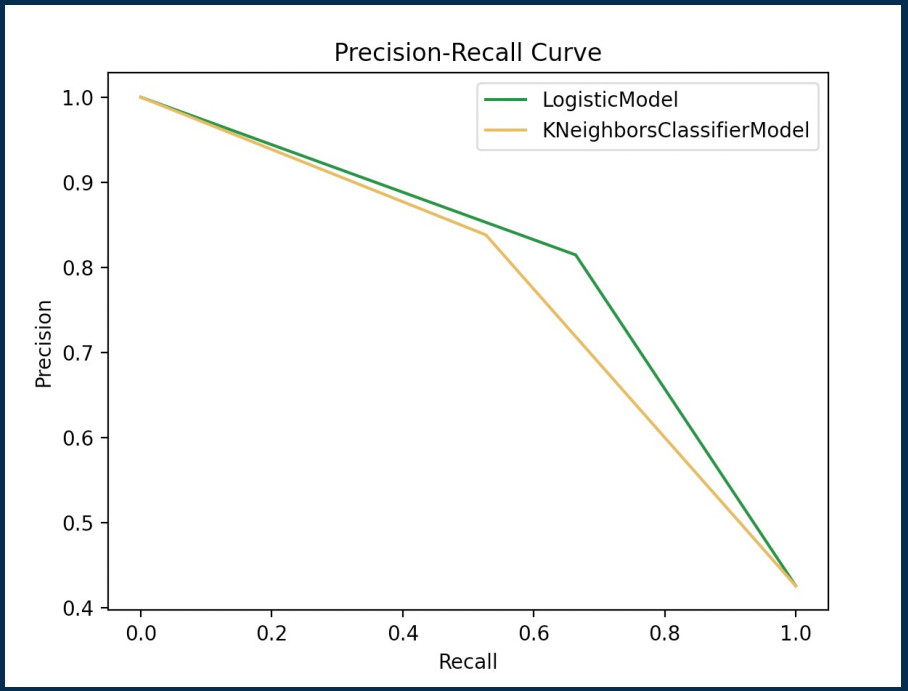
This model implements the k-nearest neighbor's vote.

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.7317487266553348}
Elapsed: 0.0932 seconds
```

```
Results for KNeighborsClassifierModel:

{'accuracy': 0.7550886408404465,
'precision': 0.8382352941176471,
'recall': 0.5269645608628659,
'f1_score': 0.6471144749290444}
Elapsed: 0.3079 seconds
```



Naïve Bayes

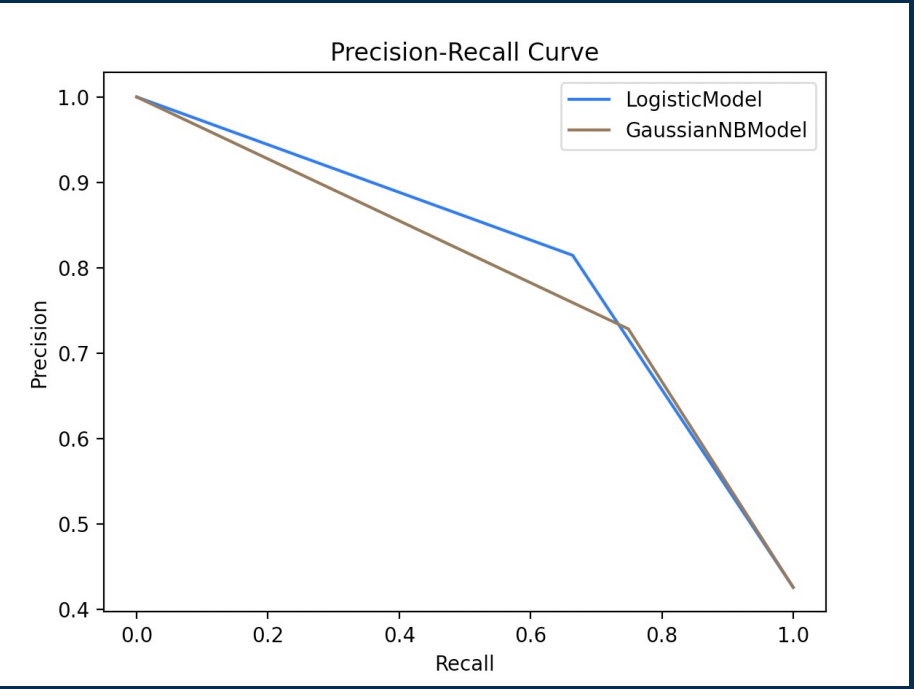
This model has a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.7317487266553 48}
Elapsed: 0.0932 seconds

Results for GaussianNBModel:

{'accuracy': 0.7741300065659882,
'precision': 0.7286356821589205,
'recall': 0.74884437596302,
'f1_score': 0.7386018237082066}
Elapsed: 0.4472 seconds
```



Gradient Boosting Classifier

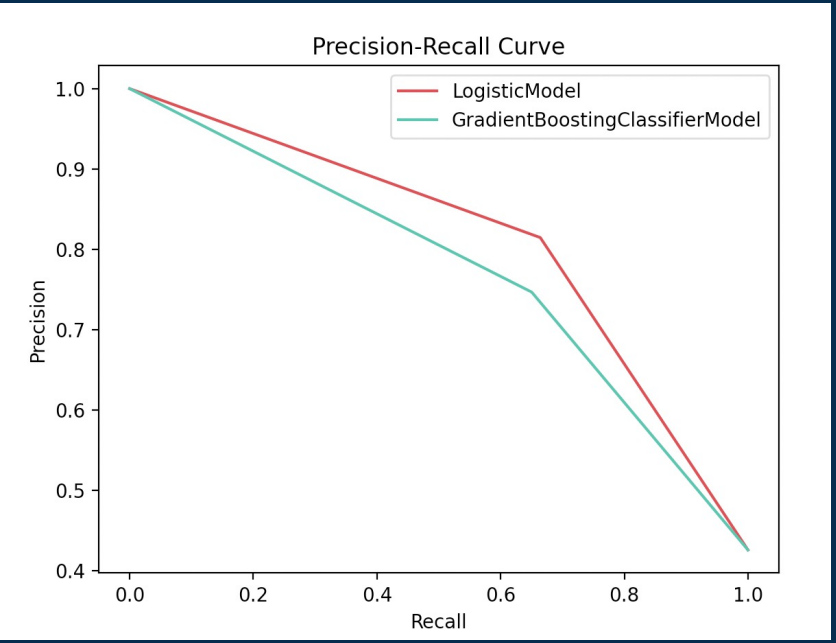
This model builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions.

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.731748726655348}
Elapsed: 0.0932 seconds


Results for GradientBoostingClassifierModel:

{'accuracy': 0.7570584372948129,
'precision': 0.7469026548672566,
'recall': 0.650231124807396,
'f1_score': 0.6952224052718287}
Elapsed: 1.014 seconds
```



Support Vector Classifier (SVC)

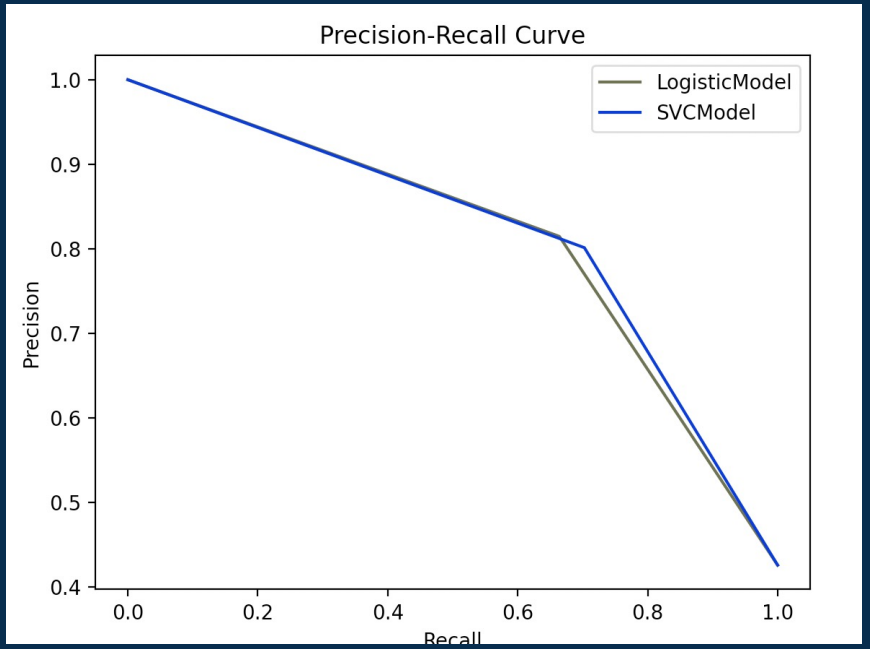
This model is a linear support vector classification model.

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.731748726655348}
Elapsed: 0.0932 seconds
```

```
Results for SVCModel:

{'accuracy': 0.799080761654629,
'precision': 0.8014059753954306,
'recall': 0.7026194144838213,
'f1_score': 0.7487684729064038}
Elapsed: 0.1691 seconds
```



Multi-layer Perceptron (MLP) Classifier

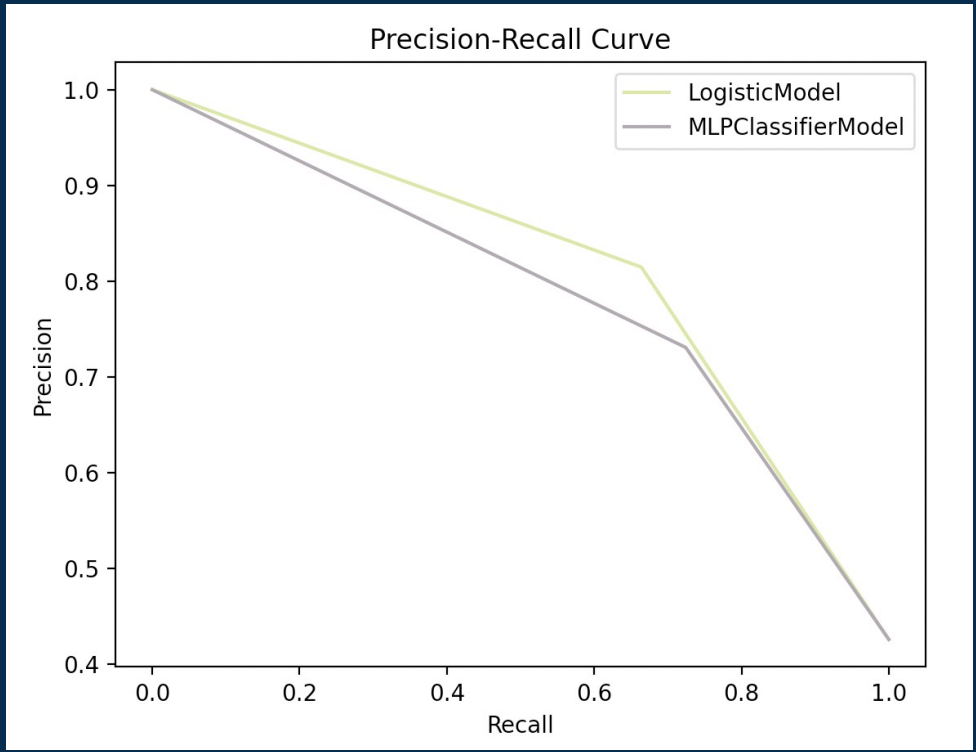
This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

```
Results for LogisticModel:

{'accuracy': 0.7925147734734077,
'precision': 0.8147448015122873,
'recall': 0.6640986132511556,
'f1_score': 0.731748726655348}
Elapsed: 0.0932 seconds
```

```
Results for MLPClassifierModel:

{'accuracy': 0.7688772160210111,
'precision': 0.7309486780715396,
'recall': 0.724191063174114,
'f1_score': 0.7275541795665633}
```



Source Code

Source Code

- https://github.com/tahirbags/cs410_project_public/tree/main

Leaderboard Competition Creation using Natural Language Processing with disaster tweets dataset

Team: "HAAAT" for CS410 Group Project

Members: *H*arish Venkata (hkv2), *A*brielle Agron (aa106), *A*ryan Gandhi (aryang6), *A*shwin Saxena (ashwins2), *T*ahir Bagasrawala (tahirib2)

README file inspired by this previous project.

Demo Video Link <! --- update link and video presentation -->

Project Description

Leaderboard Competition evaluates several generative and descriptive classifiers against Kaggle's Natural Language Processing with Disaster Tweets to predict which tweets are about real disasters and which are not.

This project falls under Theme: Leaderboard Competition.

Project Structure and Source Code

Leaderboard Competition soure code consists of following:

- /docs: Documentation for final project presentation, and project progress status report
- /models: 11 total models (incl. baseline), a combination of discriminative and generative classifiers, were developed, tuned and tested against the dataset.
    i. **AdaBoostClassifer.py**: AdaBoost classifier is a 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.

```
if model_to_tune == 'MLP':
    mlp_model = MLPClassifier(max_iter=10000)

    parameter_space = {
    'hidden_layer_sizes': [(10,30,10),(20,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.05, 0.01],
    'learning_rate': ['constant','adaptive'],
    }

elif model_to_tune == 'AdaBoost':
    """
    List of base_estimators below:

    AdaBoostClassifier,
    BernoulliNB,
    DecisionTreeClassifier,
    ExtraTreeClassifier,
    ExtraTreesClassifier,
    MultinomialNB,
    NuSVC,
    Perceptron,
    RandomForestClassifier,
    RidgeClassifierCV,
    SGDClassifier,
    SVC.
    """
    mlp_model = AdaBoostClassifier()
```