AMRITA VISHWA VIDYAPEETHAM

SENTIMENTAL ANALYSIS

Subject : Machine Learning

Subject Code : 19CSE305

Kakavakam Jaswanth Sai

CH.EN.U4CS420130

INDEX

PROBLEM STATMENT

- Nowadays, the **opinion or review** of the person plays a significant part in the sales of the product or the state of the problem while we are discussing any topic or buying any goods.

- The largest profit we can make comes from the sentiment expressed in that opinion or review. Manually determining the sentiment, whether it be **positive, negative, or neutral**, takes a lot of effort; thus, we may employ Text mining algorithms to quickly do a SENTIMENTAL ANALYSIS.

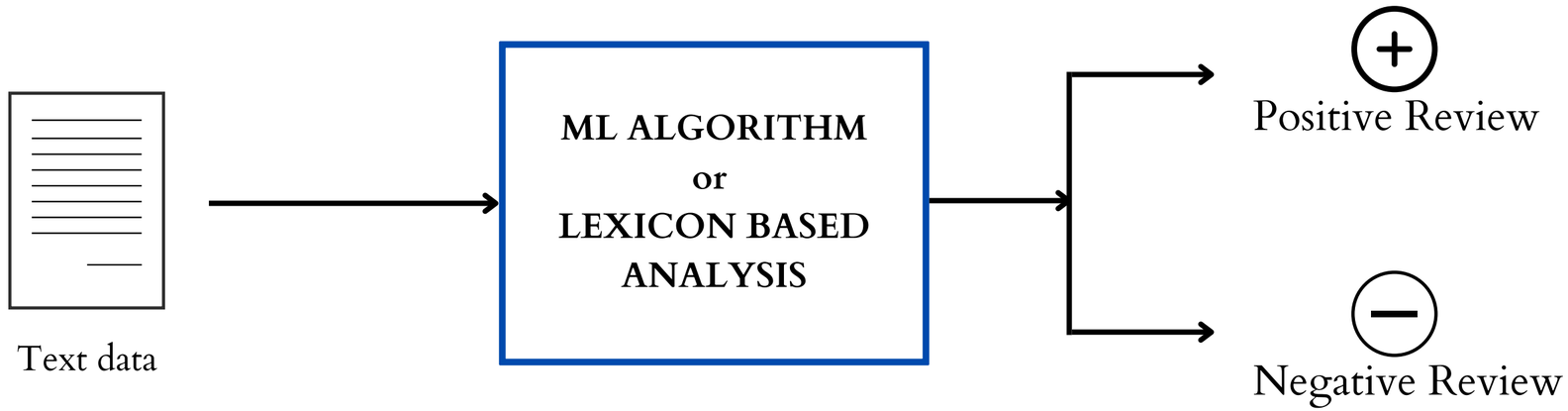
LITERATURE SURVEY

Year	Method	Domain	Accuracy	Author
2009	Mixed ML algorithams	Product reiviews	83.30	E. Boiy and M.-F. Moens
2011	Lexicon Based	Movie reiviews	76.37	J. Brooke, M. Tofiloski and M. Taboada
2013	RNN (ML)	Movie reiviews	85.40	R. Socher, A. Perelygin, J.Y. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts et al.,
2015	PRDM (DL)	Movie reiviews	86.50	C. Li, B. Xu, G. Wu, S. He, G. Tian and Y. Zhou
2017	SVM, RF, NN, Bagging	Weighted Fuzzy rule-based SA-Tweets	73.2,78.9,65.9	Syed Muzamil Basha, Yang Zhenning , Dharmendra Singh Rajput*, Iyengar N.Ch.S.N and Ronnie D. Caytiles
2017	SVM	Twitter Sentiment Analysis on Demonetization tweets	72%	K.Arun * 1 , A.Srinagesh **2 , M.Ramesh**3
2020	Navie bayes +SVM	Esports & education curiculum	66.92%	K. Sentamilselvan, D. Aneri, A. C. Athithiya, P. Kani Kumar

SOLUTION

- There are various **algorithms** available in machine learning which can do the job of classifying the given text review as positive, negative, or neutral.
- We want to find mainly whether the given review is positive or negative means we need to do a **binary classification** of the text. Hence, we have SVM a famous binary classification algorithm.
- Before moving to **ML Algorithm** first I want to walk through the **LEXICON-BASED** analysis method to find the classification of the text.
- Proposed Solution : **StackingClassifier (SVM + Logistic + RF + NB)**

INPUTS & OUTPUT



- The input of the model will be **text** which can be anything (i.e., **review, opinion, comment**, etc..). when we push the data to the model the **expected output** is whether the given text is **positive or negative**.

DATASET DESCRIPTION

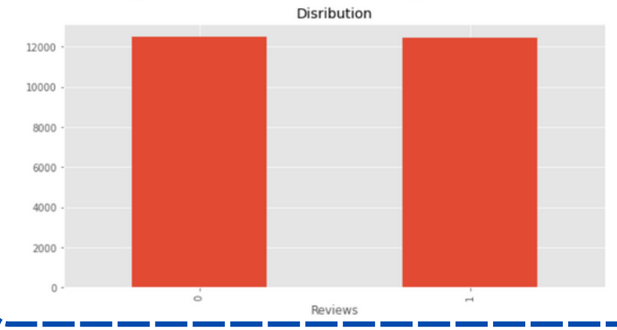
- Previously, I mentioned the Twitter dataset as my working dataset in the assessment, but the same data set was analyzed so many times.Hence I shifted to another dataset of **IMDB Reviews**.
- **IMDB Dataset.csv** is an dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing.
- In this IMDB Dataset, which is **preprocessed** in such a way it is reaching **my goal** i.e., it has only two attributes one is **Review** and the second is **Sentiment**.
- As mentioned previously it has **50k** Data Entries. This will be a good amount of data for **training and testing** my model.
- Next Regarding **Input** of My Model will be the **Text Data** which is here Review of Movies in IMDB. **Output**, Will be the **Sentiment** i.e., whether it is **Positive, Neutral & Negitive**.
- In the next slide where I am going to discuss the **Lexicon approach** which I studied there, the output will be the probability of positive, negative, and neutral is given for each review.

LEXICON BASED APPROCH

- We see the standard Vader sentimental analysis.

Preprocessing of dataset

Firstly do the **EDA** and visualize the **distribution** of the data. Also, **labelEncode the positive-1 & negitive-0.**



- Now we have the text which is not preprocessed the first step is **tokenizing.**
- We are going to use the famous **NLTK** package word tokenizing.

Parts of speech tag for each token will help us to differentiate meaning easily. Hence we use the pos_tag() function from the same NLTK package.

LEXICON BASED APPROCH

- We see the standard Vader sentimental analysis.

Preprocessing of dataset

After POS tagging the data will be in the dictionary now we need to convert that into **chunks** which are converting it into a single word. We use the **ne_chunk()** function.

Vader Sentimental Analysis

- We will download Vader(**Valence Aware Dictionary and sEntiment Reasoner**) lexicon dictionary. which helps to analyze.
- Using SentimentIntensityAnalyzer function of **nltk.sentiment package** we will predict the **polarity_score** of the processed text.
- **polarity_score()** function provides us with the probability of **negative, neutral, positive, and compound** of the given text.
- With a **threshold value** on compound probability, we can classify the text to positive and negative sentiment.

LEXICON BASED APPROCH

Threshold value selection

- Normally we can take the threshold as 0 where
  - compound probability > 0 can be classified as **Positive Text.**
  - compound probability <= 0 can be classified as **negative Text.**

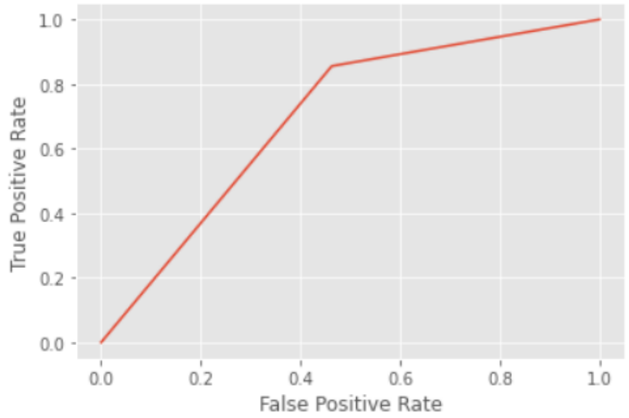
```
from sklearn.metrics import classification_report
print(classification_report(vaders["Sent_vale"], r))
```

	precision	recall	f1-score	support
0	0.79	0.54	0.64	25000
1	0.65	0.86	0.74	25000
accuracy			0.70	50000
macro avg	0.72	0.70	0.69	50000
weighted avg	0.72	0.70	0.69	50000

```
from sklearn.metrics import f1_score,accuracy_score,precision_score
print("Accuracy score : ",accuracy_score(vaders["Sent_vale"],r)*100)
print("Precision score : ",precision_score(vaders["Sent_vale"],r)*100)
print("F1 score : ",f1_score(vaders["Sent_vale"],r, average='weighted')*100)
```

```
Accuracy score :  69.632
Precision score :  64.88987318730659
F1 score :  68.84191261282973
```

Model ROC CURVE :



LEXICON BASED APPROCH

Threshold value selection

- If we see the accuracy is not that good(**69.6**) hence we need to examine it with different threshold values.

- By calculating the accuracy with different threshold values I have come up with an optimal threshold value of 0.8 Where the accuracy is 72.258 which is the highest compared to all other threshold values.

```
from sklearn.metrics import classification_report
print(classification_report(vaders["Sent_vale"], r))
```

```
              precision    recall  f1-score   support

           0       0.72      0.72      0.72     25000
           1       0.72      0.72      0.72     25000

    accuracy                           0.72     50000
   macro avg       0.72      0.72      0.72     50000
weighted avg       0.72      0.72      0.72     50000
```

Threshold value	Accuracy	Presicion	F1 score
-0.2	69.054	64.083	68.059
-0.1	69.368	64.525	68.492
0	69.632	64.889	68.841
0.1	69.886	65.243	69.171
0.2	70.17	65.694	69.551
0.3	70.509	66.254	69.995
0.4	70.852	66.871	70.440
0.5	71.2	67.651	70.906
0.6	71.676	68.668	71.490
0.7	71.99	69.9	71.912
0.8	72.258	71.767	72.254
0.9	71.492	74.986	71.351

Is 72.2 good accuracy ??

- **Yes**, it is not good one hence we need to find some other way to classify the text.
- Hence, the **LEXICON-based approach** is not that reliable we will move to **machine learning techniques.**
- Now I will run the dataset with some famous **binary classifiers** and decide which to be considered or should **ensemble** it.
- SVM, Logistic Regression, Naive Bayes, and Random forest are the algorithm I will be using.

Data Preprocessing

- The **preprocessing** of text we followed in the lexicon-based approach will not be fit for sending data in the **ML technique**.
- I have taken a whole dataset of **50000** data for my whole training and testing. **25000** are positive and **25000** and negatively classified sentiments. Also, I have checked for null values as there is **no null value** in the data.I have moved to Text preprocessing.
- Before moving to remove the noise in text data we will change the sentiment column from categorical to numerical with **LableEncoder() where positive -> 1 and negative ->0**.
- Firstly, Using the **tweet-preprocessing package** of python I have done the **basic cleaning** there after i have declared some **regular expressions** to clean the data.

Data Preprocessing

- After cleaning the data I split the data in to train and test using the **train_test_split()** function.
- The next step is **CountVectorizer** which helps us to **Convert a collection of text documents to a matrix of token counts.** with parameters **binary to true** and **stopword to English.**

	Hello	James	hello	is	james	my	name
0	0	0	1	1	1	1	1
1	1	1	0	1	0	1	1

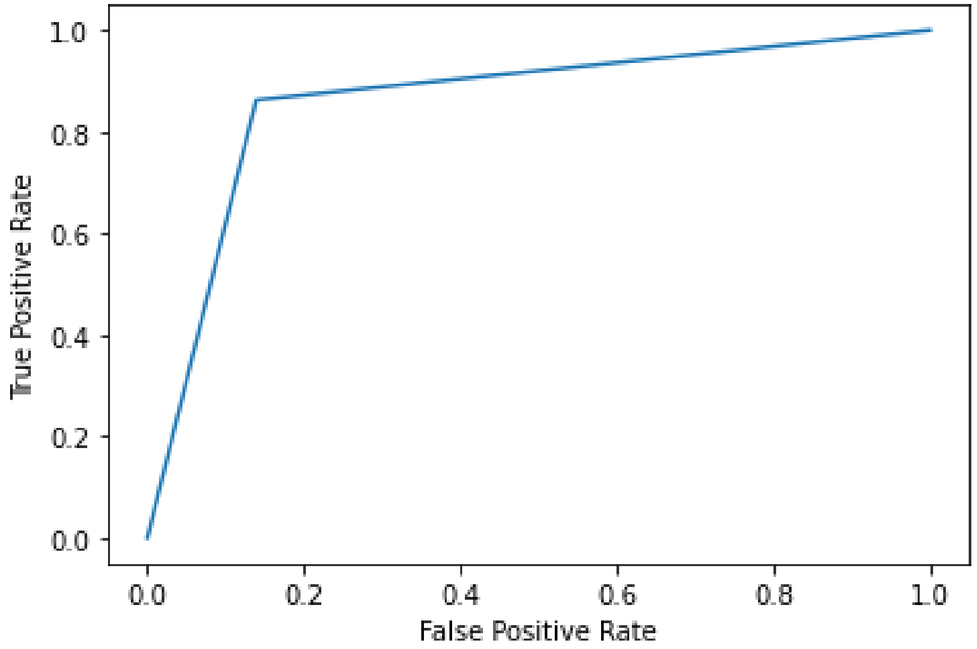
```
text = ['hello my name is james',
'Hello my name is James']
coun_vect = CountVectorizer(lowercase=False)
count_matrix = coun_vect.fit_transform(text)
count_array = count_matrix.toarray()
```

SVM – Linear

- Using LinearSVC function from sci-kit learn package I trained and tested the data

Performance Evaluation

```
Model performance for Training set
- Accuracy: 100.0
----------------------------------
Model performance for Test set
- Accuracy: 86.17333333333333
----------------------------------
- Precision: 86.38342789286185
----------------------------------
- F1-Score: 86.32647679324896
----------------------------------
Model performance for Cross-validation
- Accuracy: 86.00571428571429
----------------------------------
```

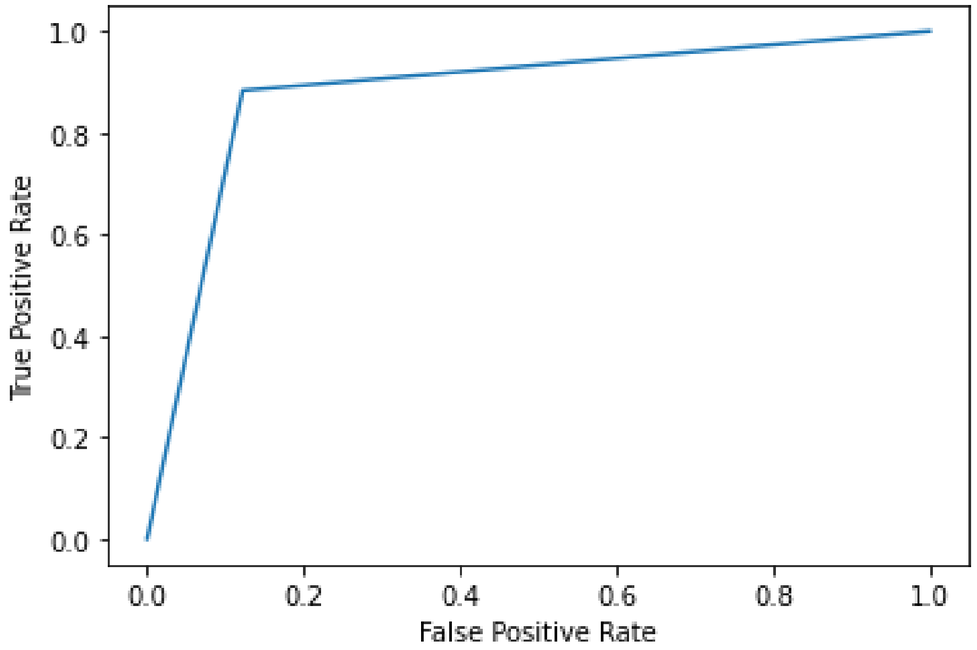


Logistic Regression

- Using LogisticRegression() function from sci-kit learn package I trained and tested the data

Performance Evaluation

```
Model performance for Training set
- Accuracy: 99.70857142857142
----------------------------------
Model performance for Test set
- Accuracy: 88.06
----------------------------------
- Precision: 88.11464633184329
----------------------------------
- F1-Score: 88.21322803553801
----------------------------------
Model performance for Cross-validation
- Accuracy: 87.63714285714286
----------------------------------
```

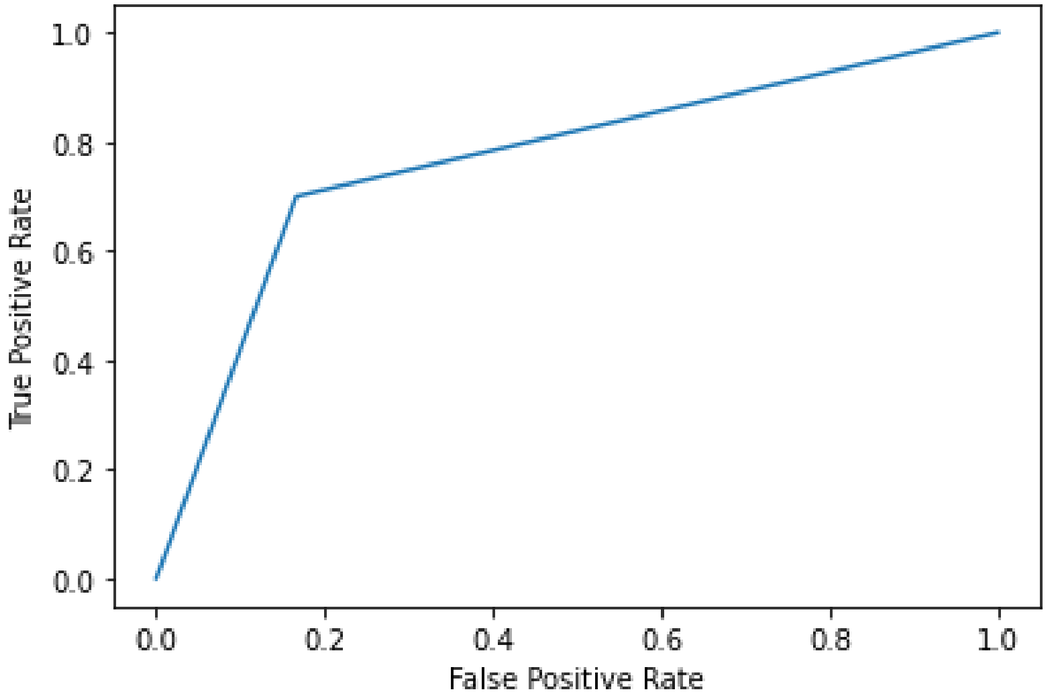


Random Forest

- Using Random Forest() function from sci-kit learn package I trained and tested the data

Performance Evaluation

```
Model performance for Training set
- Accuracy: 99.44285714285715
----------------------------------
Model performance for Test set
- Accuracy: 76.6
----------------------------------
- Precision: 81.15167252176569
----------------------------------
- F1-Score: 75.169779286927
----------------------------------
Model performance for Cross-validation
- Accuracy: 76.04571428571428
----------------------------------
```

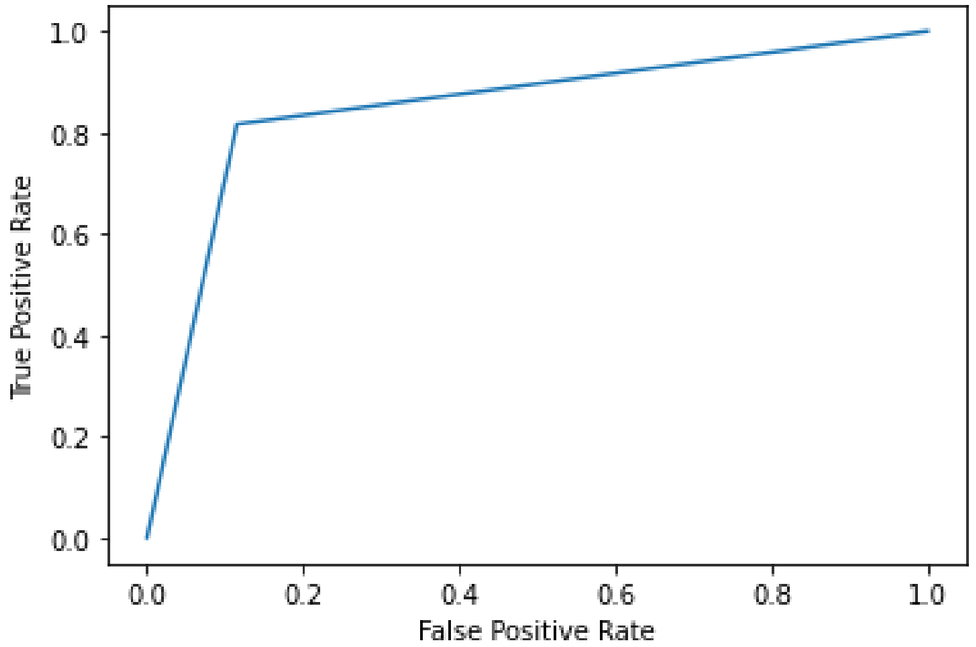


NB CLASSIFIER

- Using BernoulliNB() function from sci-kit learn package I trained and tested the data

Performance Evaluation

```
Model performance for Training set
- Accuracy: 91.02285714285713
----------------------------------
Model performance for Test set
- Accuracy: 85.02
----------------------------------
- Precision: 87.89727582292849
----------------------------------
- F1-Score: 84.64849354375896
----------------------------------
Model performance for Cross-validation
- Accuracy: 84.51142857142857
----------------------------------
```

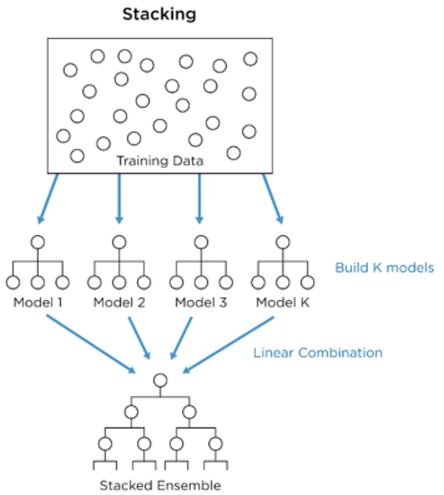


- As we see four accuracies is near 80% for the test data but here if we see one has 88% and one has 86% and the other has 76% to get good accuracy out of all these learning model we need another method.

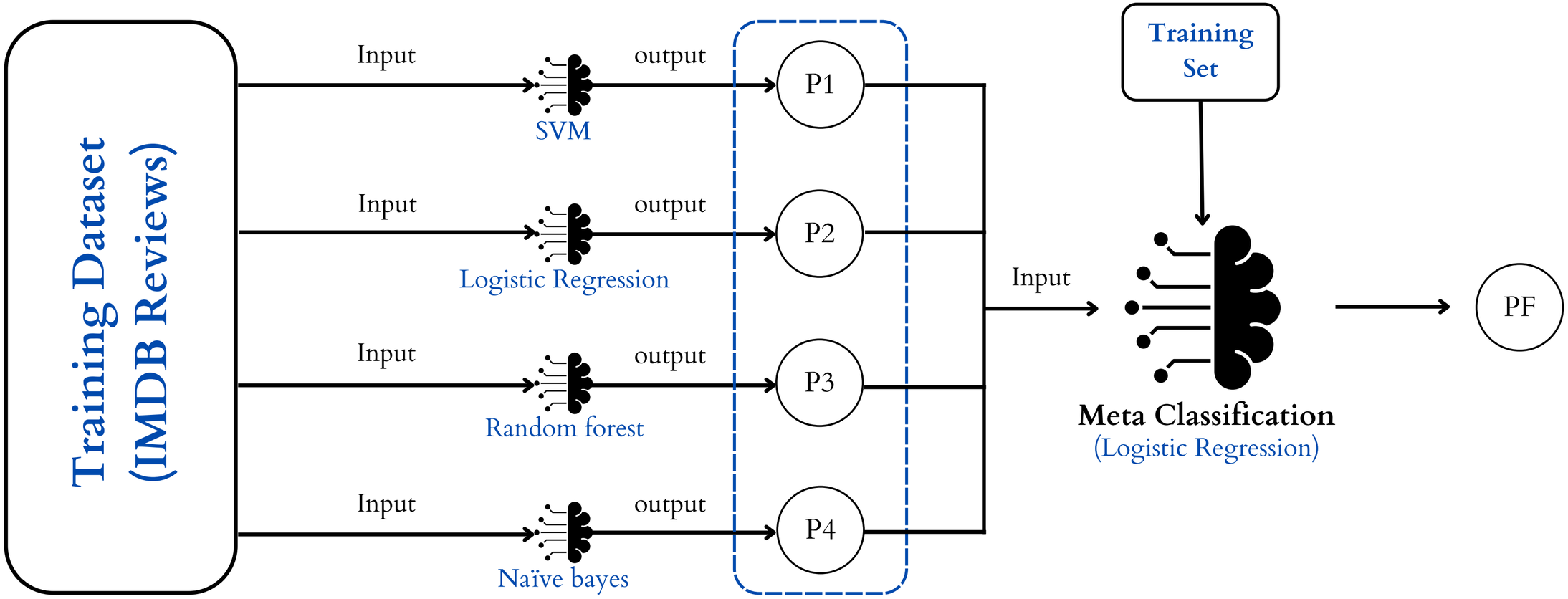
ENSEMBLE METHOD

↓

StackingClassifier



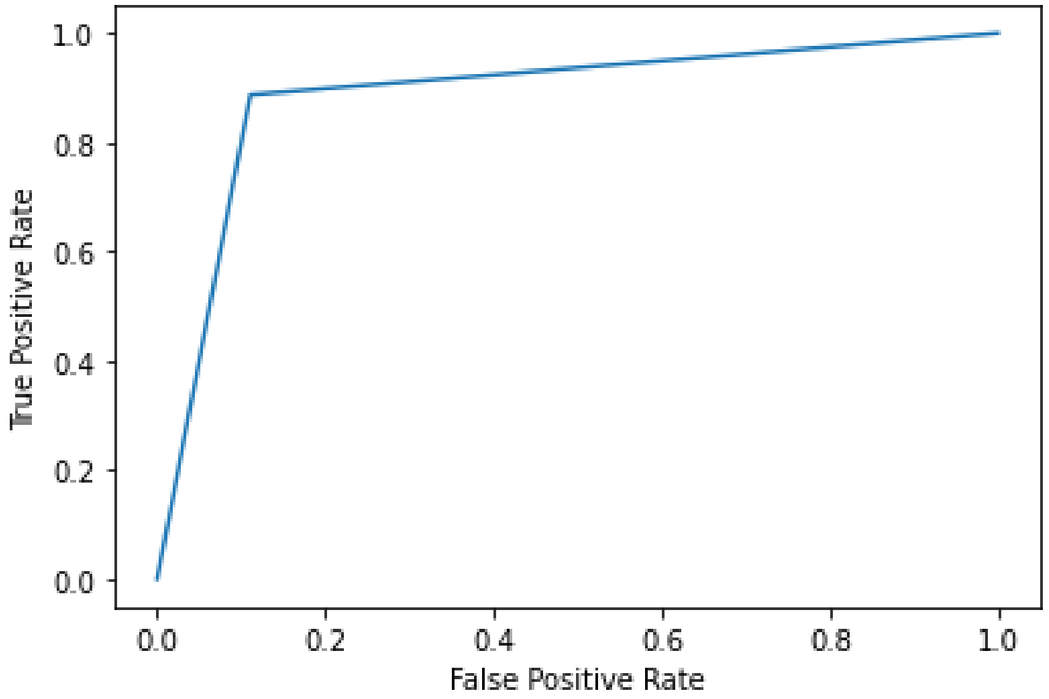
StackingClassifier – Architecture



StackingClassifier

- **Performance Evaluation**

```
Model performance for Training set
- Accuracy: 99.72285714285715
----------------------------------
Model performance for Test set
- Accuracy: 88.8
----------------------------------
- Precision: 89.12726791153489
----------------------------------
- F1-Score: 88.90356671070013
----------------------------------
Model performance for Cross-validation
- Accuracy: 88.3714285714285 8
----------------------------------
```



Comparitive Study

	Accuracy_train	Accuracy_test	Precision_test	F1_score_test
svm_linear	100.000000	86.173333	86.383428	86.326477
rf	99.442857	76.600000	81.151673	75.169779
bnb	91.022857	85.020000	87.897276	84.648494
logistic	99.708571	88.060000	88.114646	88.213228
stack	99.722857	88.800000	89.127268	88.903567

Future scope & Conclusion

- After working with all 4 models I have come up with an **ensemble stacking model** as my final working model.
- Which is working with the **best accuracy(88.8 ~ 89%)** and the model **Roc curve** is also in the range of the best model area.
- Possible future work in this area is adding **Lexicon-based and ML algorithm together** which may leads to a high level of accuracy.

AMRITA VISHWA VIDYAPEETHAM

Thank You

Jaswanth Sai kakavakam | CH.EN.U4CSE20130

jaswanthsaikakavakam@gmail.com