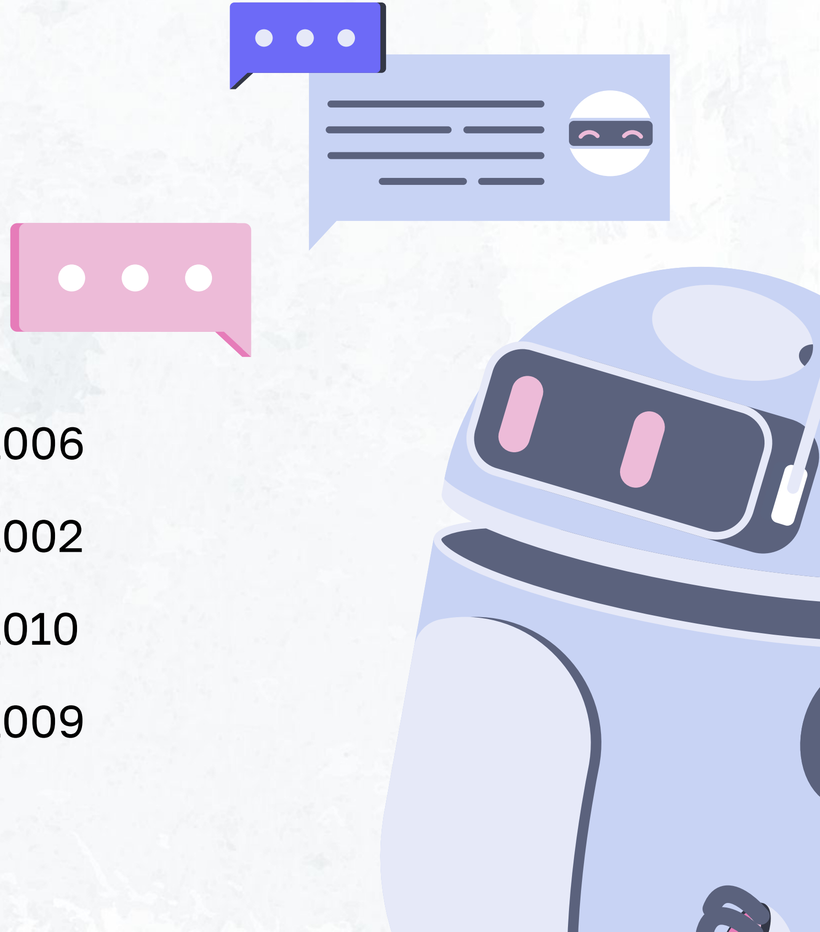
IMDB Reviews Sentiment Prediction



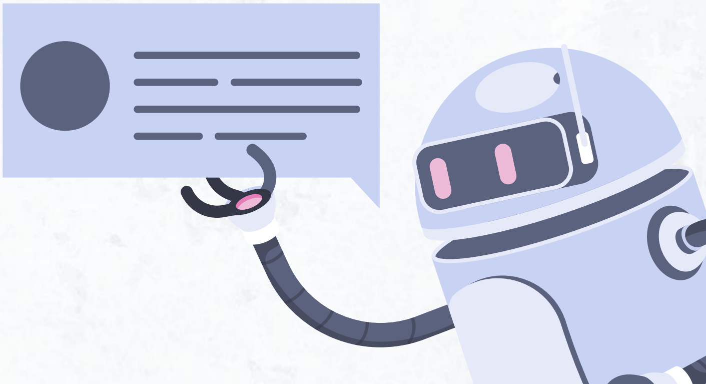
Ang Woei Haw	MCS221006
Ho Zhi Hoong	MCS221002
Mehdi Mouaiz Eddine Smail	MCS221010
Rheshwan Raj A/L Ravichandran	MCS221009

Executive summary

The aim is to perform sentiment analysis on the IMDB Dataset of 50K Movie Reviews obtained from Kaggle. The dataset consists of movie reviews and corresponding sentiment labels (positive or negative). The project involves several steps, starting with data preprocessing to remove duplicates and convert sentiment labels into binary form. Text normalization techniques are applied to clean the reviews, including the removal of HTML tags and URLs, expanding contractions, and converting text to lowercase. Tokenization is then performed to eliminate punctuation, stopwords, and perform stemming. Exploratory data analysis (EDA) is conducted to gain insights into the dataset, including the creation of word cloud and visualizations. Common words are removed from the dataset to improve model performance. The dataset is split into training and testing sets, and TF-IDF (Term Frequency-Inverse Document Frequency) is applied to convert text into numerical features. Three supervised learning algorithms - logistic regression, random forest, and linear SVM - are trained on the data using both the modified and unmodified data. The models' performance is evaluated and compared using confusion matrices.

(-) Project Overview →

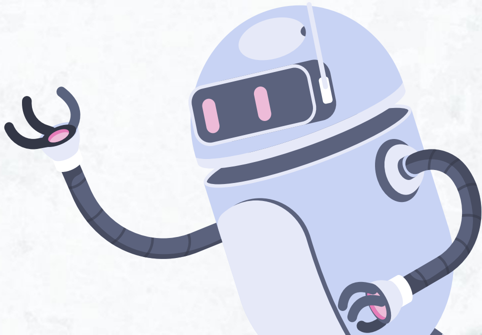
- Our project focuses on sentiment analysis of the IMDB Dataset of 50K Movie Reviews from Kaggle.
- The dataset consists of movie reviews accompanied by sentiment labels (positive or negative).



Problem Statement →

- Our objective is to classify movie reviews accurately based on their sentiment using machine learning methods.

- By analyzing the sentiment of movie reviews, we can gain valuable insights into audience perceptions and preferences, contributing to decision-making processes in the film industry.



Significance

- Sentiment analysis holds significant importance in various domains, including marketing, customer feedback analysis, and brand reputation management.

- Our project offers a comprehensive analysis of sentiment in movie reviews, providing insights into the overall perception of movies and assisting filmmakers, production companies, and movie enthusiasts in understanding audience opinions.

- By leveraging machine learning methods and comparing the results of Random Forest, Logistic Regression, and Support Vector Machines (SVM), we can evaluate the performance of different approaches, identifying the strengths and weaknesses of each method in sentiment analysis.

Data

IMDB Dataset of 50K Movie Reviews

Kaggle's IMDB Dataset of 50K Movie Reviews is a dataset for binary sentiment classification. The dataset contains 50,000 movie reviews and their corresponding sentiments, which are categorized as positive or negative. The dataset contains two columns: "movie review" and "sentiment." The written content of the reviews is provided by the "movie review" column, which captures individuals' thoughts, views, and reactions to the films they watched. The "sentiment" column indicates whether the reviewer's sentiment is positive or negative, and it reflects the reviewer's overall emotional tone or perspective.

review	sentiment
49582 unique values	2 unique values
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The...	positive
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet	negative

Data Preprocessing

```python
# remove dulplicate data
data.drop_duplicates(subset='review', keep='first', inplace=True)

# covert 1 for "postive" and 0 for neagtive
data['target'] = data["sentiment"].apply(lambda i: 1 if i == "positive" else 0)
```

```python
data['sentiment'].value_counts()
```

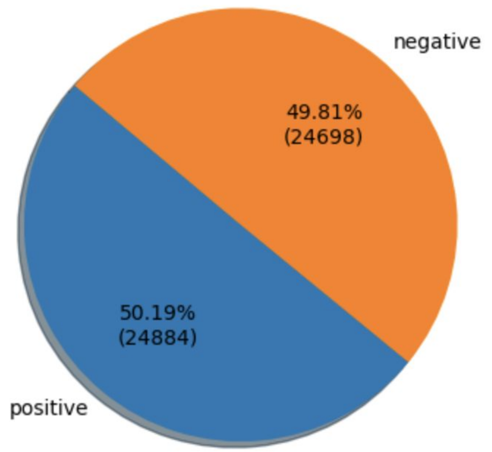
```
positive    24884
negative    24698
Name: sentiment, dtype: int64
```

```python
data
```

	review	sentiment	target
0	One of the other reviewers has mentioned that ...	positive	1
1	A wonderful little production. The...	positive	1
2	I thought this was a wonderful way to spend ti...	positive	1
3	Basically there's a family where a little boy ...	negative	0
4	Petter Mattei's "Love in the Time of Money" is...	positive	1
...	...	...	...
49995	I thought this movie did a down right good job...	positive	1
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative	0
49997	I am a Catholic taught in parochial elementary...	negative	0
49998	I'm going to have to disagree with the previou...	negative	0
49999	No one expects the Star Trek movies to be high...	negative	0

49582 rows × 3 columns

```python
# Dataset visualization
def my_fmt(x):
    return '{:.2f}%\n({:.0f})'.format(x, total*x/100)
sentiment_count = data["sentiment"].value_counts()
total = len(data)
plt.pie(sentiment_count, labels=sentiment_count.index,
        autopct=my_fmt, shadow=True, startangle=140)
plt.show()
```



Text Normalization

Processing	**Initial Text**	**Processed Text**
Lowercase	**A**n **A**merican actress **M**arigold, played by **A**li **C**arter 33 gets stuck in **I**ndia	**a**n **a**merican actress **m**arigold, played by **a**li **c**arter 33 gets stuck in **i**ndia
Remove HTML tags	** **anyway please tell me if you agree or disagree with me	anyway please tell me if you agree or disagree with me
Remove URL	One can also find good reviews regarding this movie at **http://www.comingsoon.net/films.php?id=36310**	One can also find good reviews regarding this movie at
Expand Contraction	LOOOOOOOOK at this ... **i'd** like it so much!!!!!	LOOOOOOOOK at this ... **I would** like it so much!!!!!
Tokenization	an american actress marigold, played by ali carter 33 gets stuck in india	['an', 'american', 'actress', 'marigold', ',', 'played', 'by', 'ali', 'carter', '33', 'gets', 'stuck', 'in', 'india']
Remove punctuation	['an', 'american', 'actress', 'marigold', **','**, 'played', 'by', 'ali', 'carter', '33', 'gets', 'stuck', 'in', 'india']	['an', 'american', 'actress', 'marigold', 'played', 'by', 'ali', 'carter', '33', 'gets', 'stuck', 'in', 'india']
Remove stopwords	[**'an'**, 'american', 'actress', 'marigold', 'played', **'by'**, 'ali', 'carter', '33', 'gets', 'stuck', **'in'**, 'india']	['american', 'actress', 'marigold', 'played', 'ali', 'carter', '33', 'gets', 'stuck', 'india']
Keep alphabet only	['american', 'actress', 'marigold', 'played', 'ali', 'carter', **'33',** 'gets', 'stuck', 'india']	['american', 'actress', 'marigold', 'played', 'ali', 'carter', 'gets', 'stuck', 'india']
Stemming	['american', 'actress', 'marigold', **'played'**, 'ali', 'carter', **'gets'**, 'stuck', 'india']	['american', 'actress', 'marigold', **'play'**, 'ali', 'carter', **'get'**, 'stuck', 'india']

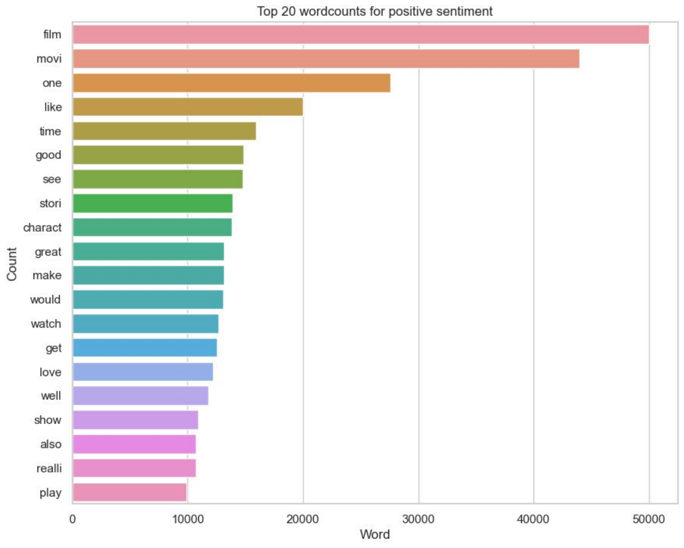
Exploratory Data Analysis

```
# token words appear in postive sentiment
pos_tweets = data[data["sentiment"] == "positive"]
text = "".join(tweet.lower()for tweet in str(pos_tweets["token"]))
wordcloud = WordCloud(background_color='green').generate(text)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
# token words appear in postive sentiment
neg_tweets = data[data["sentiment"] == "negative"]
text = "".join(tweet.lower()for tweet in str(neg_tweets["token"]))
wordcloud = WordCloud(background_color='red').generate(text)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

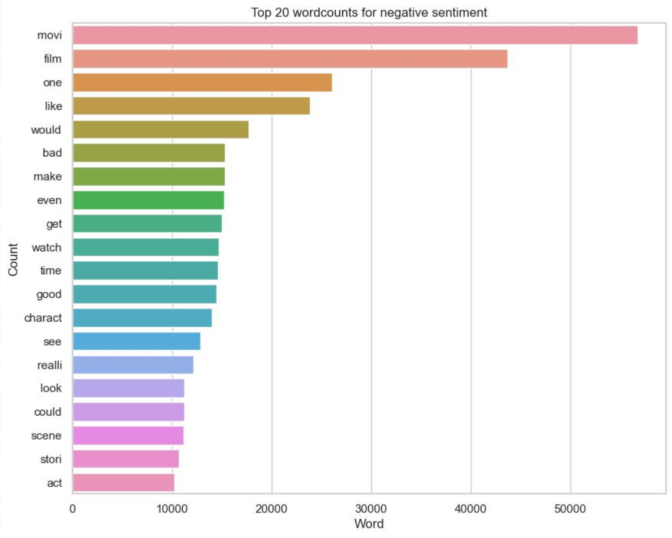
```python
wordcount_plot(postive_word, "positive")
```

	Word	Count
0	film	49987
1	movi	43978
2	one	27547
3	like	20016
4	time	15942
5	good	14819
6	see	14814
7	stori	13922
8	charact	13826
9	great	13168
10	make	13165
11	would	13120
12	watch	12715
13	get	12565
14	love	12236
15	well	11797
16	show	10918
17	also	10718
18	realli	10697
19	play	9908




```python
wordcount_plot(negative_word,"negative")
```

	Word	Count
0	movi	56798
1	film	43743
2	one	26087
3	like	23863
4	would	17712
5	bad	15356
6	make	15334
7	even	15236
8	get	15034
9	watch	14689
10	time	14666
11	good	14457
12	charact	14056
13	see	12861
14	realli	12200
15	look	11283
16	could	11235
17	scene	11145
18	stori	10746
19	act	10262



Removing common words in positive and negative comments

```python
# from the EDA part some words appear very frequent in both sentiment
# like 'movi','film','one','like'
# try remove these word , to check is it help to improve performance

def remove_word (token):
    common_word = ["movi", "film","one","like"]
    token_new = token.copy()
    #print(token_new)
    for word in token_new:
        if word in common_word:
            token_new.remove(word)
    return token_new
```

```python
data["token_new"] = data["token"].apply(remove_word)
```

```python
wordcount_plot(postive_word, "positive")
```

	Word	Count
0	film	49987
1	movi	43978
2	one	27547
3	like	20016
4	time	15942
5	good	14819
6	see	14814
7	stori	13922
8	charact	13826
9	great	13168
10	make	13165
11	would	13120
12	watch	12715
13	get	12565
14	love	12236
15	well	11797
16	show	10918
17	also	10718
18	realli	10697
19	play	9908

```python
wordcount_plot(negative_word,"negative")
```

	Word	Count
0	movi	56798
1	film	43743
2	one	26087
3	like	23863
4	would	17712
5	bad	15356
6	make	15334
7	even	15236
8	get	15034
9	watch	14689
10	time	14666
11	good	14457
12	charact	14056
13	see	12861
14	realli	12200
15	look	11283
16	could	11235
17	scene	11145
18	stori	10746
19	act	10262

Text Vectorization

7 Train Test Split

```
# for token
X = data["token"]
y = data["target"]
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size=0.3)

# token_new with removed 'common words'
X_removed = data["token_new"]
X_train_re , X_test_re , y_train_re , y_test_re = train_test_split(X_removed , y , test_size=0.3)
```

8 TF-IDF

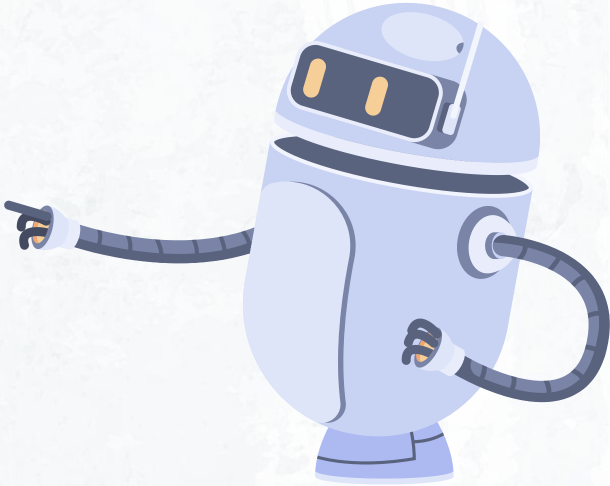
```
tfid = TfidfVectorizer(use_idf=True)
def fit_tfidf(tweet_corpus):
    tf_vect = TfidfVectorizer(preprocessor = lambda x :x,
                              tokenizer = lambda x : x)
    tf_vect.fit(tweet_corpus)
    return tf_vect
```

```
# for token
tf = fit_tfidf(X_train)
X_train_tf = tf.transform(X_train)
X_test_tf = tf.transform(X_test)

# # for token_new
tf_re = fit_tfidf(X_train_re)
X_train_tf_re =  tf_re.transform(X_train_re)
X_test_tf_re = tf_re.transform(X_test_re)
```

```
print(X_train_tf)
```

```
  (0, 56475)	0.16906751489746913
  (0, 56044)	0.1371580000987352
  (0, 55431)	0.09369836938713844
  (0, 53998)	0.11973946029402621
  (0, 53441)	0.07883842228690578
  (0, 52670)	0.14084596218197254
  (0, 50408)	0.0857242247750621
  (0, 47936)	0.04630637266667537
  (0, 47758)	0.11817830248193312
  (0, 46181)	0.08363393448912292
  (0, 46168)	0.1708920714719192
  (0, 45023)	0.19499803824108833
  (0, 44460)	0.07479756021972632
  (0, 44068)	0.10323692834718667
  (0, 43732)	0.04988047242560254
  (0, 42341)	0.06569755035553478
  (0, 42189)	0.18330530918113874
  (0, 41887)	0.1496533440273918
  (0, 40820)	0.11772008531659282
  (0, 40206)	0.11629548488548891
  (0, 40172)	0.186359086339866
  (0, 39765)	0.14352391010803245
  (0, 39444)	0.07067994452049992
  (0, 39326)	0.13536095927855424
  (0, 38611)	0.16202524277833774
  :	:
  (34706, 4207)	0.021532735033299284
  (34706, 3687)	0.06783392318589544
  (34706, 3602)	0.043351688497320504
  (34706, 3327)	0.030279772866111106
  (34706, 2908)	0.04560949673925213
  (34706, 2856)	0.02439055066730488
  (34706, 2684)	0.15334141812519253
  (34706, 2649)	0.026039569722628858
```

Random Forest

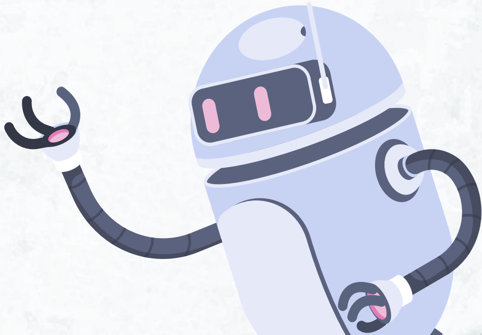


Overview

- Random Forest is an ensemble learning method that combines multiple decision trees to make predictions.
- It is a popular algorithm for classification and regression tasks due to its robustness and ability to handle large datasets.
- Random Forest creates an ensemble of decision trees, where each tree is trained on a random subset of features and a subset of the training data.
- During prediction, the output is determined by aggregating the predictions of all the individual trees.

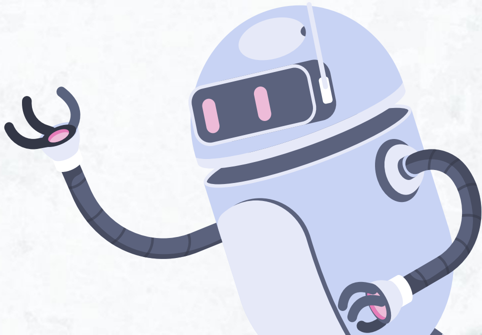
(-) Advantages Of Random Forest

- Random Forest can handle a large number of input features without overfitting the model.
- It provides estimates of feature importance, allowing for better understanding of the underlying data.
- The algorithm is less sensitive to outliers and can handle missing values.



(-) Advantages Of Random Forest →

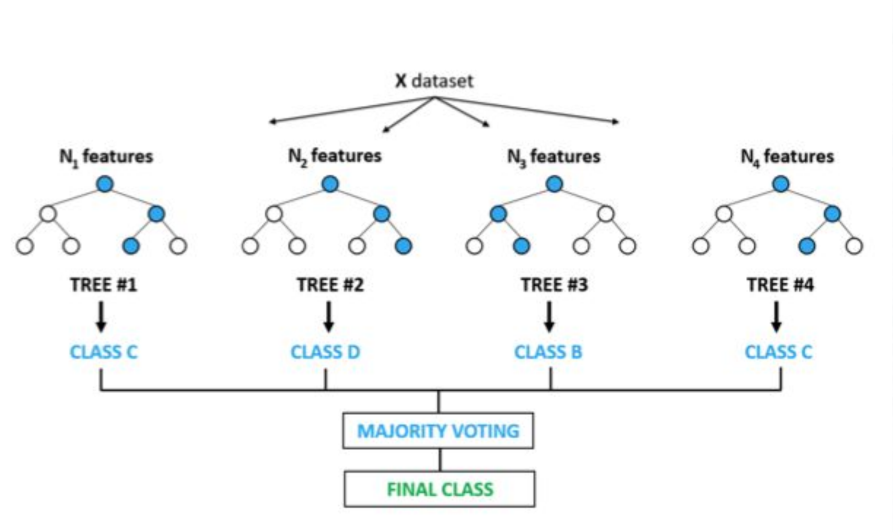
- Random Forest can handle a large number of input features without overfitting the model.
- It provides estimates of feature importance, allowing for better understanding of the underlying data.
- The algorithm is less sensitive to outliers and can handle missing values.

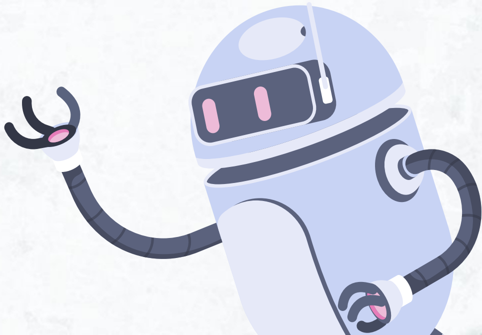


(-) Random Forest in Sentiment Analysis

- Random Forest can be effectively used for sentiment analysis tasks, including classifying movie reviews as positive or negative.

- By leveraging the ensemble of decision trees, Random Forest can capture complex relationships and improve the accuracy of sentiment predictions.





Implementation (Old Token)

- Common words are not remove

```
# Random Forest Model
random_forest_model = RandomForestClassifier()
random_forest_model.fit(X_train_tf, y_train)
```

```
y_pred_tf_random_forest = random_forest_model.predict(X_test_tf)
```

```
rf_accu = accuracy_score(y_test, y_pred_tf_random_forest)*100
rf_recall = recall_score(y_test, y_pred_tf_random_forest)*100
rf_precision = precision_score(y_test, y_pred_tf_random_forest)*100
print("Random Forest Model Accuracy: {:.2%}".format(accuracy_score(y_test, y_pred_tf_random_forest)))
print("Random Forest Model Recall Score: {:.2%}".format(recall_score(y_test, y_pred_tf_random_forest)))
print("Random Forest Model Precision Score: {:.2%}".format(precision_score(y_test, y_pred_tf_random_forest)))
print(classification_report(y_test, y_pred_tf_random_forest))
```

Implementation (New Token)

- Common word are remove

```
random_forest_model.fit(X_train_tf_re, y_train_re)

y_pred_tf_re = random_forest_model.predict(X_test_tf_re)

print("Random Forest Model Accuracy: {:.2%}".format(accuracy_score(y_test_re, y_pred_tf_re)))
print("Random Forest Model Recall Score: {:.2%}".format(recall_score(y_test_re, y_pred_tf_re)))
print("Random Forest Model Precision Score: {:.2%}".format(precision_score(y_test_re, y_pred_tf_re)))
print(classification_report(y_test_re, y_pred_tf_re))
```

Performance

Common word are not removed

```
Random Forest Model Accuracy: 85.06%
Random Forest Model Recall Score: 84.97%
Random Forest Model Precision Score: 85.17%
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      7425
           1       0.85      0.85      0.85      7450

    accuracy                           0.85     14875
   macro avg       0.85      0.85      0.85     14875
weighted avg       0.85      0.85      0.85     14875
```

Common word removed

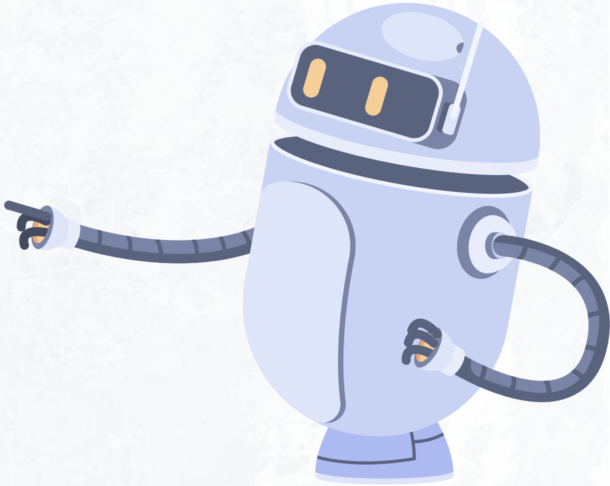
```
Random Forest Model Accuracy: 85.06%
Random Forest Model Recall Score: 84.97%
Random Forest Model Precision Score: 85.17%
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      7425
           1       0.85      0.85      0.85      7450

    accuracy                           0.85     14875
   macro avg       0.85      0.85      0.85     14875
weighted avg       0.85      0.85      0.85     14875
```

Misclassification

```
Review : ['When i first saw the movie being advertised i thought it was going to be another Disney movie that almos
t goes straight to video. I finally got around and rented it. I thought it was going to be bad because i couldn\'t
see Shia in any other role than his recently cancelled show "Even Stevens". When i turned it on i was ready to turn
it off from boredom in about ten minutes. It started a bit slow and i couldn\'t understand the beginning because th
e years didn\'t make sense then they explained that later in the show so i was relieved of wondering about that. Al
l and all i thought it was a good movie and i would recommend it. The cast was top notch and even though i\'m not a
fan of golf it easily kept my attention with a good plot.']

Sentiment : ['positive']

Predicted Sentiment :negative
```

```
Review : ['The female cast of this movie is terrific: you\'ve got Linda Blair (maturing nicely), Julie Strain (who
doesn\'t get too many speaking lines - that\'s a good thing), Rochelle Swanson (equally convincing as a sweet innoc
ent girl or as an evil possessed girl), Toni Naples, and the most beautiful of them all IMO, the simply stunning Kr
istina Ducati (how the goofy male lead, Larry Poindexter, deserved to get sexually involved with any of these women
remains a mystery). However, beyond the chance to watch these beautiful and in some cases talented women, the movie
has little to offer. The plot is disjointed and doesn\'t really get going until the last 15 minutes or so; and when
Wynorski finally manages to create some suspense, a ludicrous "twist" ending comes and ruins everything. (*1/2)']

Sentiment : ['negative']

Predicted Sentiment :postive
```

Linear Support Vector Machine



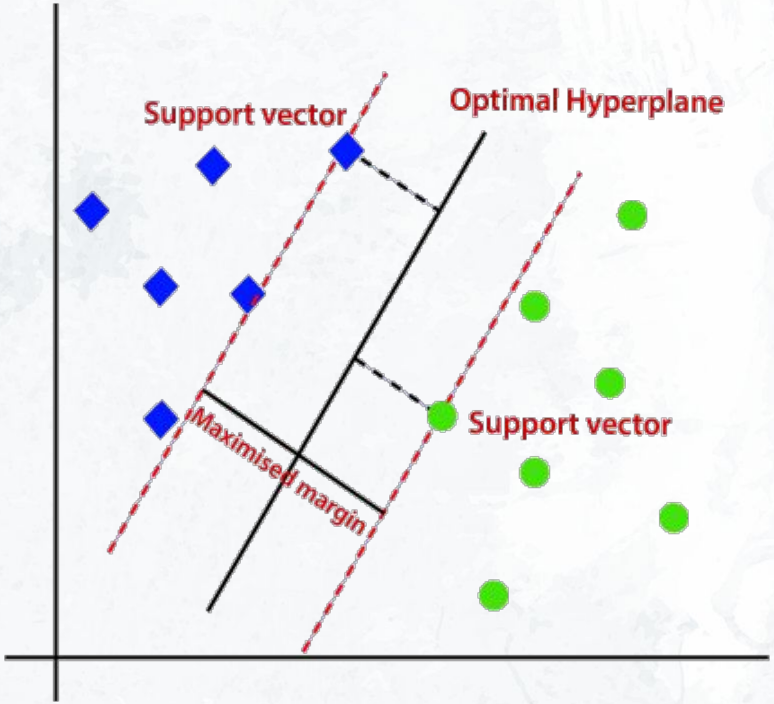
Overview

Used to predict binary outcome based on prior examination of the dataset

Uses linear equation ***(w^t * x + b)*** to create the largest possible margin between 2 classes (to find the hyperplane)

'w' represents weight vector, ***'x'*** feature vector of data point, ***'b'*** bias term and ***'w^t'*** denotes the transpose of weight vector

Example of application
handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages



Implementation

- Common words are not removed

```
# Create an instance of the SVM classifier
svm_model =  LinearSVC(loss='hinge')
# Train the SVM model
svm_model.fit(X_train_tf, y_train)
```

```
# Make predictions
y_pred_tf_svm = svm_model.predict(X_test_tf)

print("SVM Model Precision Score: {:.2%}".format(precision_score(y_test, y_pred_tf_svm)))
print("SVM Forest Model Recall Score: {:.2%}".format(recall_score(y_test, y_pred_tf_svm)))
print("SVM Forest Model Accuracy: {:.2%}".format(accuracy_score(y_test, y_pred_tf_svm)))

print(classification_report(y_test, y_pred_tf_svm))

plot_confusion(confusion_matrix(y_test, y_pred_tf_svm));
```

Implementation

- Common words are removed

```
svm_model.fit(X_train_tf_re, y_train_re)

y_pred_tf_re_svm = svm_model.predict(X_test_tf_re)
print(classification_report(y_test_re, y_pred_tf_re_svm))
```

Performance

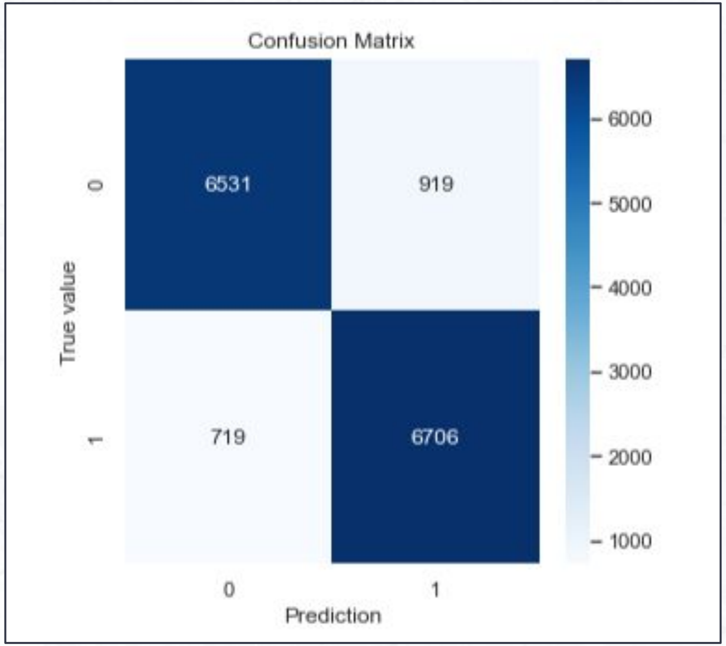
	precision	recall	f1-score	support
0	0.90	0.88	0.89	7382
1	0.88	0.90	0.89	7493
accuracy			0.89	14875
macro avg	0.89	0.89	0.89	14875
weighted avg	0.89	0.89	0.89	14875

Common words are not removed

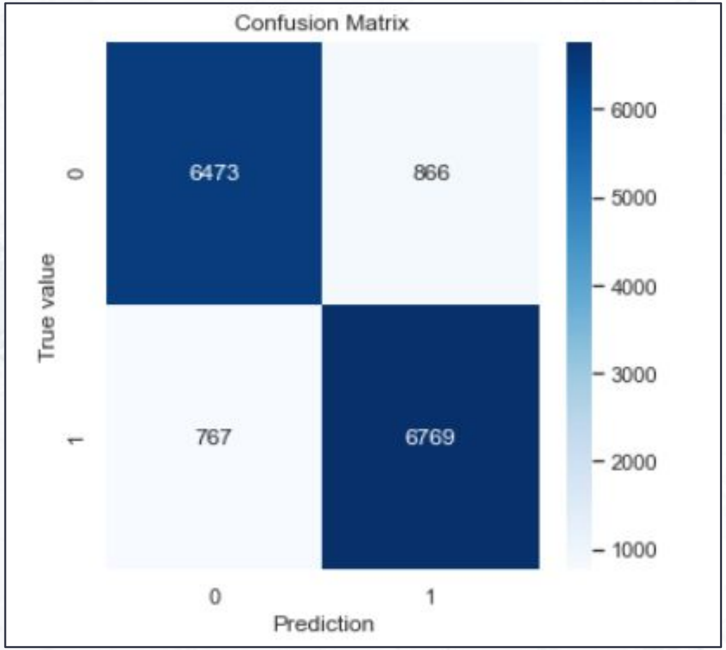
	precision	recall	f1-score	support
0	0.89	0.88	0.89	7339
1	0.89	0.90	0.89	7536
accuracy			0.89	14875
macro avg	0.89	0.89	0.89	14875
weighted avg	0.89	0.89	0.89	14875

Common words are removed

Performance



Common words are not removed



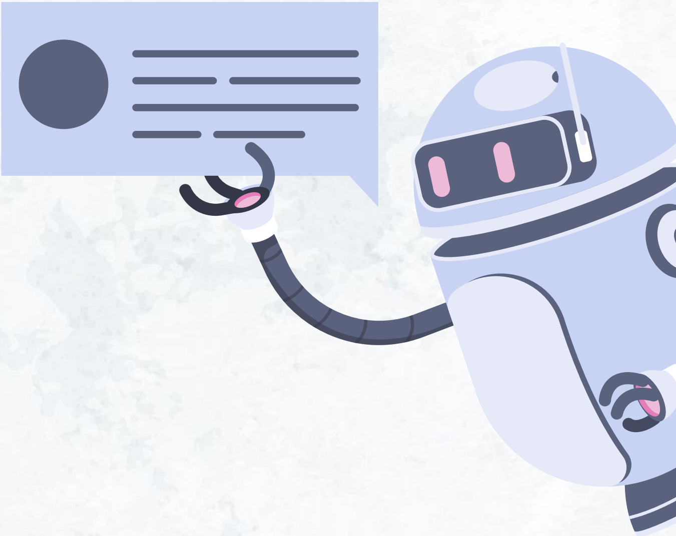
Common words are removed

Misclassification

```
Review (SVM): ['It\'s probably a year since I saw Uzak, but it has left strong memories of the two main characters, jaded photo
grapher Mahmut and his naive cousin from the village Yusuf.  It\'s a long film with very little dialogue and a quite limited pl
ot. This has evidently annoyed a fair few viewers. But the film constructs such a painfully believable portrait of Mahmut and Y
usuf that there\'s just as much emotional tension as in the paciest thriller.  Just to be clear, there\'s no padding in this fi
lm -- in the long pauses where no one speaks there as much happening in the characters\' emotions (and in yours, watching them)
as you could bear. Go to see it awake and alert, and you\'ll be gripped rather than anaesthetised.  Uzak rings true in so many
ways, and that sincerity is probably its greatest accomplishment. People don\'t grapple with events and problems, so much as wi
th each other. In fact, in the whole film, there\'s probably not one point where the main characters (Mahmut, Yusuf and Mahmut
\'s ex-wife Nazan) are not opposed.  Much of it is true the world over: country cousin Yusuf\'s perhaps wilfully naive expectat
ion that a job on a ship will drop into his lap; Mahmut\'s urbanised cynicism and unwillingness to sympathise with Yusuf.  Othe
r truths are more-specific to Turkey: Yusuf\'s incomprehension that Mahmut might be tolerating his stay with gritted teeth; Yus
uf veering between macho ambition and wide-eyed awkwardness when he tries to get to know a woman.  Uzak is undoubtedly a pretty
bleak film, and one Ceylan\'s strengths is not to beat us over the head with the themes he explores. For me at least, I believe
d entirely in the behaviour of his characters. All the little failed attempts to connect and petty cruelties ring so true. And
yet I didn\'t leave with a message that "The world is like that", but instead I got "This is how we sometimes treat each othe
r."']

Sentiment: ['positive']

Predicted Sentiment (SVM): negative
```

```
Review (SVM): ["But quite dated today. Otto Preminger made this movie without the certificate of approval that was needed then.
It was enormously courageous and risky as he could have lost his investment and future.  The film is not true to the wonderful
book and is unfortunately hollywoodized.  Frank Sinatra (and I've never been a fan) playing Frankie Machine, is astonishing in
his performance. One forgets it is Frank up there, the level of realism he brings to the role of a jonesing drug addict has to
be seen to be believed.  Kim Novak, eternally gorgeous and talented, does not disappoint in the role of the devoted outsider, a
lways there for Frankie.  Supporting roles, particularly a young, handsome and talented Darrin Mc Gavin, are faultless.  Eleano
r Parker, playing Frankie's wife, is hopelessly inept. She swings from irritating to melodramatic and is far too over the top.
A forgettable performance.  The stagey, cheap settings are appalling, as if a firm gust of wind would blow the whole tacky pain
ted cardboards over the horizon. Almost laughable at times in their tawdry cheapness.  The music was irritating, poundingly so
at times. As if each nuance of the script (example: when Louie is getting Frankie his fix out of a drawer) had to be underscore
d at a high decibel level.  7 out of 10. Sinatra truly deserved his Oscar nomination. Worth seeing."]

Sentiment: ['positive']

Predicted Sentiment (SVM): negative
```

Logistics Regression

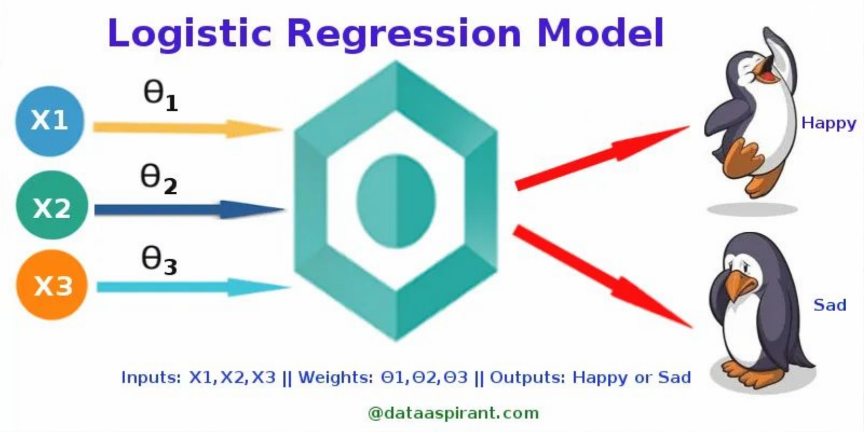


Overview

Used to predict binary outcome (Yes or No) based on prior observation of dataset.

Sigmoid function is the hypothesis of Logistics Regression - ranges from 0 to 1. The threshold is usually set to 0.5 . If the value is greater than 0.5 the outcome is 1 otherwise 0 .

Example of application -   Predicting spam email and fraudulent transaction



Implementation

- Common words are not remove

```
model = LogisticRegression()
model.fit(X_train_tf, y_train)
```

```
y_pred_tf = model.predict(X_test_tf)

print("Logistics Regression Model Accuracy: {:.2%}".format(accuracy_score(y_test, y_pred_tf)))
print("Logisitic Regression Model Recall Score : {:.2%}".format(recall_score(y_test, y_pred_tf)))
print("Logistoc Regression Model Precision Score : {:.2%}".format(precision_score(y_test, y_pred_tf)))
print(classification_report(y_test,y_pred_tf))
```

Implementation

- Common word are remove

```
model.fit(X_train_tf_re, y_train_re,)
```

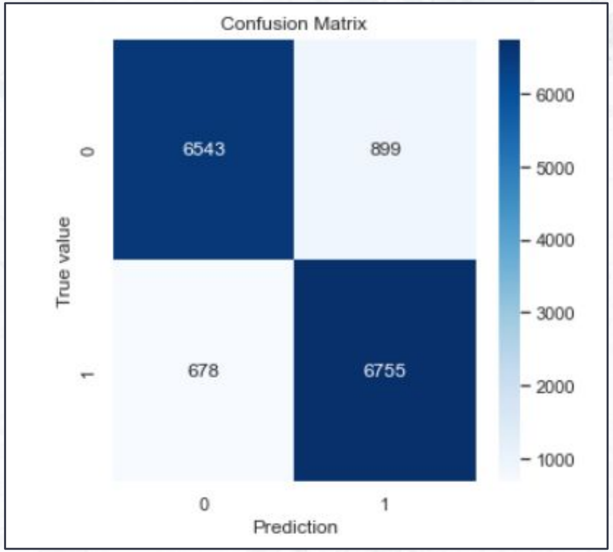
```
y_pred_tf_re = model.predict(X_test_tf_re)
print("Logistics Regression Model Accuracy: {:.2%}".format(accuracy_score(y_test_re, y_pred_tf_re)))
print("Logisitic Regression Model Recall Score : {:.2%}".format(recall_score(y_test_re, y_pred_tf_re)))
print("Logistoc Regression Model Precision Score : {:.2%}".format(precision_score(y_test_re, y_pred_tf_re)))
print(classification_report(y_test_re,y_pred_tf_re))
```

Performance

- Common word are not removed

Logistics Regression Model Accuracy: 89.40%
Logisitic Regression Model Recall Score : 90.88%
Logistoc Regression Model Precision Score : 88.25%

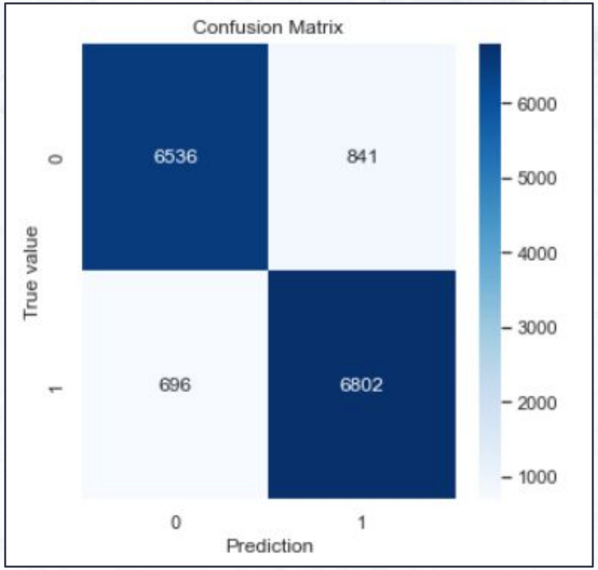
	precision	recall	f1-score	support
0	0.91	0.88	0.89	7442
1	0.88	0.91	0.90	7433
accuracy			0.89	14875
macro avg	0.89	0.89	0.89	14875
weighted avg	0.89	0.89	0.89	14875



Performance

- Common Word are removed

```
Logistics Regression Model Accuracy: 89.67%
Logisitic Regression Model Recall Score : 90.72%
Logistoc Regression Model Precision Score : 89.00%
              precision    recall  f1-score   support

           0       0.90      0.89      0.89      7377
           1       0.89      0.91      0.90      7498

    accuracy                           0.90     14875
   macro avg       0.90      0.90      0.90     14875
weighted avg       0.90      0.90      0.90     14875
```



Misclassification

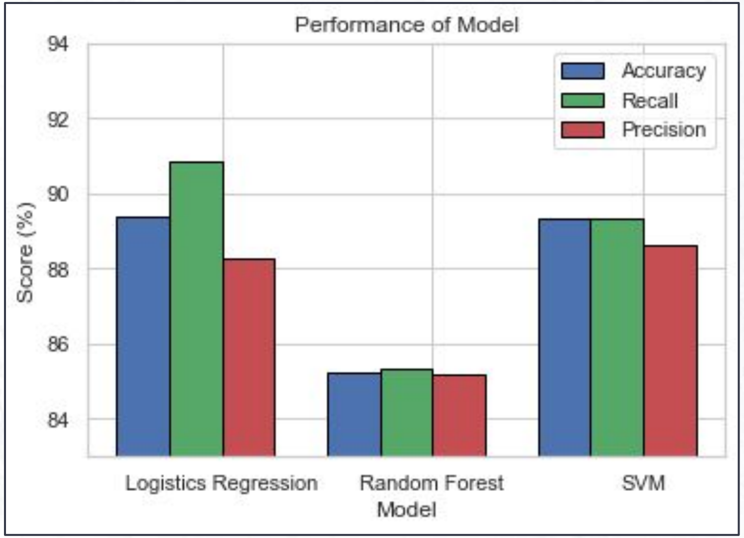
```
Review : ["I must say I was disappointed with this film. Although it is well acted and directed, the underlying story simply pl
ods along too slowly.  Granted, in another mood I would have liked it better. I did chuckle a lot, but rarely laughed out loud;
and there was actually a sense of suspense to discover who won. But in contrast to another movie that my wife picked up the sam
e day (one neither of us had heard of before) this one paled in comparison.  If you see lots of movies, then by all means see t
his -- it's distinctly better than your average fare. But if you (like me) have limited time and want to watch only the best an
d most entertaining, save this for later.  [Rate: 7/10]"]

Sentiment : ['negative']

Predicted Sentiment :postive
```

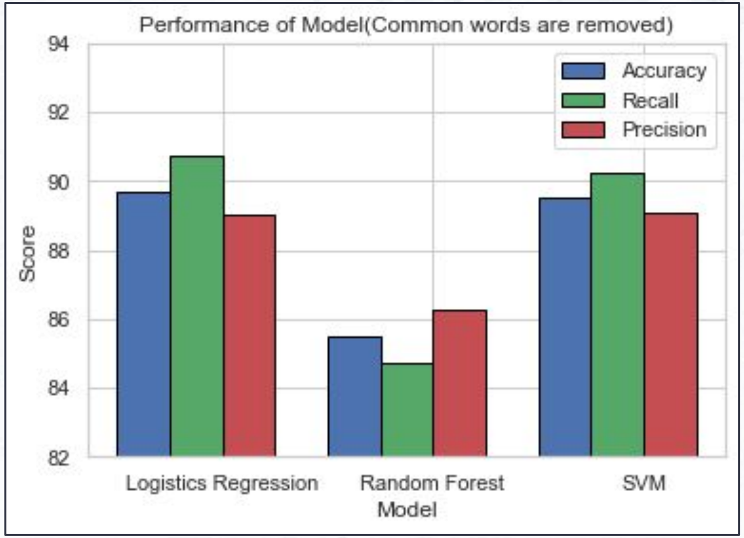
```
Review : ["This was a surprisingly very good movie, and an interesting idea.. However, it was just a little bit disappointing i
n that the 'Twist' was a little too predictable and just a bit too early on in the movie. Whilst watching, it started to get a
little bizarre and confusing to the point that, the only reasonable outcome possible was the inevitable plot twist, but it cert
ainly did not ruin this movies flow. There were superb performances, there was never a dull boring moment, so totally well wort
h watching this one. It kept me interested right up until the end, and for me there isn't many movies that can do that these da
ys. I highly recommend people watch this terrific little movie."]

Sentiment : ['positive']

Predicted Sentiment :negative
```

Result and Findings

Model Comparison

- SVM and Logistics Regression have similar performance , while the Random Forest has the worst performance in all the evaluation metrics





Model Comparison

- The performance of each model (common words are not removed)

	Accuracy	Recall	Precision
Logistics Regression	89.40%	90.88%	88.25%
Random Forest	85.24%	85.43%	85.16%
SVM	88.63%	90.23%	89.34%

Model Comparison

- The performance of each model (common words are removed)

	Accuracy	Recall	Precision
Logistics Regression	89.67%	90.72%	89.00%
Random Forest	85.49%	84.72%	86.25%
SVM	89.50%	90.06%	90.25%

Conclusion

- Logistics regression and SVM have the similar performance on the all three evaluation metrics .

- the performance of the Random Forest is the worst among the three models.

- All three models achieve a least 85% of score for all three evaluation metrics.

- These trained models are useful for the movie or film producers and members of public.

- The film producers can use these result especially the reviews from the negative sentiments to improve their film’s quality.

- While the members of public can use the outcome of the model to know whether a film or a movie is worth to watch

Work Distributions

Parts	Ho Zhi Hoong	Mehdi Mouaiz Eddine Smail	Rheshwan Raj A/L Ravichandran	Ang Woei Haw
Executive Summary			✓	
Introduction		✓		
Data Preprocessing and Text Normalization	✓			
EDA & Text Vectorization	✓			
Machine Learning		✓ Random Forest	✓ Linear SVM	✓ Logistic Regression
Results & Discussion				✓