Project Managemen Data dan Informasi

Indonesian Hate Speech Analysis on Twitter/X After the Second Presidential Debate in 2024

Table of Contents

Abstract

This analysis seeks to investigate the nature and prevalence of hate speech in some of the tweets that trended after the second Indonesian Presidential Debate in 2024 on Twitter/X. Utilizing various machine learning models, including K-Nearest Neighbors (KNN), Logistic Regression, SVM, Naive Bayes, Decision Tree, and Random Forest, this research aims to find out whether tweets that spike in popularity after being posted are mostly hateful or not. By carefully deploying these models against a dataset of trending tweets, this research aims to determine whether there is a significant trend toward hate and hatred in post-debate attention-grabbing discourse. These findings aim to shed light on whether trending tweets align with respectful dialogue or tend to amplify hate, thereby offering insights for readers aimed at encouraging more constructive and responsible online conversations around political events on Twitter/X.

Introduction

Hate speech constitutes direct or indirect expressions of hatred aimed at individuals or groups based on inherent characteristics, such as ethnicity, religion, disability, gender, or sexual orientation (Komnas HAM, 2015). The propagation of hate speech, especially prevalent on social media platforms, often involves the use of abusive language, which encompasses verbally or in written form conveying abusive words or phrases to others (Davidson et al., 2017).

In Indonesia, abusive language commonly stems from various unfavorable conditions, including mental disorders, sexual deviance, physical disabilities, lack of modernization, breaches of etiquette, religious prohibitions, and other unfortunate circumstances (Wijana and Rohmadi, 2010).

Twitter, a significant social media platform in Indonesia, frequently becomes a conduit for spreading hate speech (Alfina et al., 2017, 2018; Putri, 2018). During the analysis period, Twitter, referred to as X for anonymity purposes, retained its presence under the twitter.com domain. The training dataset comprised tweets gathered over approximately seven months from March 20, 2018, to September 10, 2018 (Ibrohim and Budi, 2019). This dataset aimed to capture tweet patterns resembling those from trending tweets following the second presidential debate on January 7, 2024.

1. Preparation

1.1. Import Library

The first thing to do in the preparation stage is import all the libraries needed for this analysis such as pandas for processing dataframes, numpy for processing arrays, matplotlib, seaborn, and wordcloud for visualization, libraries for managing text such as nltk, re, string, and Sastrawi, and finally the Sklearn model for machine learning purposes.

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
import warnings
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import ComplementNB, BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
warnings.filterwarnings("ignore")
```

In [11]:
```python
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Rivaldi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[11]:
```
True
```

1.2. Create a Function

After importing all the necessary libraries, the next stage is to create functions that will help with analysis, here are some of these functions.

1.2.1. Text processing function

There are several functions created to process text, such as changing text to lowercase, removing unnecessary characters such as url, rt, user, emoji, etc. Then there are also functions for removing non-alphanumeric text, normalizing *alay* words, removing stop words and excess spaces, and stemming.

In [49]:
```python
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def lowercase(text):
    return text.lower()

def remove_unnecessary_char(text):
    text = re.sub('\n',' ',text)
    text = re.sub('rt',' ',text)
    text = re.sub('user',' ',text)
    text = re.sub('url',' ',text)
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+))',' ',text)
    text = re.sub('  +', ' ', text)
    return text

def remove_unicode(text):
    text = re.sub(r'\bx[a-fA-F0-9]{2}\b', '', text)
    text = re.sub(r'\bx([a-fA-F0-9]{2})', '', text)
    return text

def remove_nonaplhanumeric(text):
    text = re.sub('[^0-9a-zA-Z]+', ' ', text)
    return text

alay_dict_map = dict(zip(df2["alay"], df2["arti"]))
def normalize_alay(text):
    return ' '.join([alay_dict_map[word] if word in alay_dict_map else word for word in text.split(' ')])

def remove_stopword(text):
    text = ' '.join(['' if word in df3.stopword.values else word for word in text.split(' ')])
    text = re.sub('  +', ' ', text)
    text = text.strip()
    return text

def stemming(text):
    return stemmer.stem(text)

def remove_extra_spaces(text):
    text = re.sub('\s+', ' ', text)
    text = text.strip()
    return text
```

In [ ]:
```python
def preprocess(text):
    text = lowercase(text)
    text = remove_nonaplhanumeric(text)
    text = remove_unnecessary_char(text)
    text = normalize_alay(text)
    text = stemming(text)
    text = remove_stopword(text)
    text = remove_unicode(text)
    text = remove_extra_spaces(text)
    return text
```

1.2.2. Confusion matrix generator function

This function was created to see the prediction results from machine learning to match whether the prediction results are accurate or not for evaluation.

In [44]:
```python
def plot_confusion_matrix(y, y_predict):
    cm = confusion_matrix(y, y_predict)
    ax = plt.subplot()
    sns.heatmap(cm, annot=True, ax=ax, fmt='d')
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix')
    ax.xaxis.set_ticklabels(['Not Hate Speech', 'Hate Speech'])
    ax.yaxis.set_ticklabels(['Not Hate Speech', 'Hate Speech'])
    plt.savefig('charts/confusion_matrix.png', bbox_inches='tight')
    plt.show()
```

1.2.3. Model trainer function

This function was created to carry out training on data using several models at once such as KNN, Logistic Regression, SVM, Complement Naive Bayes, Bernoulli Naive Bayes, Decision Tree, and Random Forest. After training, the function will return the module with the highest accuracy results while generating the prediction results in the form of a confusion matrix. Apart from that, the function will also return the evaluation results of all models in the form of a dataframe.

In [4]:
```python
def vector_model(data, category, ngram):
    X = data['Tweet'].fillna(' ')
    Y = data[category]

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)

    vector = TfidfVectorizer(ngram_range=ngram, stop_words=df3['stopword'].tolist())

    X_train_scal = vector.fit_transform(X_train)
    X_test_scal = vector.transform(X_test)
    X_test_scal_new = vector.transform(df_debat["Tweet"])

    models = {
        'KNN': KNeighborsClassifier(n_neighbors=5),
        'Logistic Regression': LogisticRegression(),
        'SVM': SVC(kernel='rbf'),
        'Complement NB': ComplementNB(),
        'Bernoulli NB': BernoulliNB(),
        'Decision Tree': DecisionTreeClassifier(criterion='entropy', min_samples_split=2, random_state=42),
        'Random Forest': RandomForestClassifier(n_estimators=105, min_samples_split=2, random_state=42)
    }

    best_accuracy = 0
    best_model = None

    scores = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}

    for model_name, model in models.items():
        model.fit(X_train_scal, Y_train)
        Y_pred = model.predict(X_test_scal)

        accuracy = accuracy_score(Y_test, Y_pred)
        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_model = model

        accuracy = accuracy_score(Y_test, Y_pred)
        precision = precision_score(Y_test, Y_pred)
        recall = recall_score(Y_test, Y_pred)
        f1 = f1_score(Y_test, Y_pred)

        scores['Model'].append(model_name)
        scores['Accuracy'].append(accuracy)
        scores['Precision'].append(precision)
        scores['Recall'].append(recall)
        scores['F1 Score'].append(f1)

    scores_df = pd.DataFrame(scores)

    if best_model:
        print(f"Best Model: {best_model} with Accuracy: {best_accuracy}")
        best_model_instance = best_model
        best_model_instance.fit(X_train_scal, Y_train)
        Y_pred_best = best_model_instance.predict(X_test_scal)
        plot_confusion_matrix(Y_test, Y_pred_best)
    else:
        print("No best model found")

    return best_model, best_accuracy, scores_df, X_test_scal_new
```

2. Data Collection

There are 4 datasets used at the beginning and loaded into 4 variables.

In [48]:
```python
#url: https://www.kaggle.com/datasets/ilhamfp31/indonesian-abusive-and-hate-speech-twitter-text?select=data.csv
df1 = pd.read_csv("dataset/tweet_hs.csv")
df2 = pd.read_csv("dataset/kamus_alay.csv")
df3 = pd.read_csv("dataset/stopwordbahasa.csv", header=None).rename(columns={0: 'stopword'})
df4 = pd.read_excel("dataset/tweet_capres_2024.xlsx")
```

2.1. Hate Speech Dataset

The first dataset consists of tweet data taken from March 20, 2018 to September 10, 2018 which have been labeled as hate speech or not.

In [13]:
```python
df1.drop("Unnamed: 0",axis=1, inplace=True)
df1.head()
```

Out[13]:

	Tweet	HS
0	- disaat semua cowok berusaha melacak perhatia...	1
1	RT USER: USER siapa yang telat ngasih tau elu?...	0
2	41. Kadang aku berfikir, kenapa aku tetap perc...	0
3	USER USER AKU ITU AKU\n\nKU TAU MATAMU SIPIT T...	0
4	USER USER Kaum cebong kapir udah keliatan dong...	1

2.2. *Alay* Dictionary Dataset

The second dataset contains *alay* terms that are often used by netizens and their meanings. This dataset aims to normalize the tweet data in the first dataset.

In [14]:
```python
df2.head()
```

Out[14]:

	Unnamed: 0	alay	arti
0	0	anakjakartaasikasik	anak jakarta asyik asyik
1	1	pakcikdahtua	pak cik sudah tua
2	2	pakcikmudalagi	pak cik muda lagi
3	3	t3tapjokowi	tetap jokowi
4	4	3x	tiga kali

2.3. Indonesian Stopword Dataset

The third dataset contains stopwords that will be matched with tweets in the first dataset to be removed.

In [15]:
```python
df3.head()
```

Out[15]:

	stopword
0	ada
1	adalah
2	adanya
3	adapun
4	agak

2.4. Tweets After the Second Presidential Debate in 2024 Dataset

The fourth dataset contains tweets taken on January 9, 2024 after the second presidential debate. The data taken is in the form of the 30 most popular tweets on that date.

In [16]:
```python
df4.head()
```

Out[16]:

	Tweet
0	"Saudara-saudara ada pula yang nyinggung-nying...
1	Allhamdulilah Jabar 80% Prabowo tapi balihonya...
2	AHAHAHAHAHAHHAHAHA TEMEN KELAS GW YANG PRO PRA...
3	BREAKING NEWS: Sumber Data Yang Disampaikan An...
4	Seorang pemimpin tidak akan pernah merendahkan...

3. Data Wrangling

3.1. Preprocessing Data

Preprocessing was carried out for the first and fourth datasets to normalize and remove unnecessary characters which could later influence the prediction results with machine learning later. Preprocessing is carried out with previously declared text processing functions.

In [17]:
```python
df1["Tweet"] = df1["Tweet"].fillna('').astype(str).apply(lambda x: preprocess(x))
df4["Tweet"] = df4["Tweet"].fillna('').astype(str).apply(lambda x: preprocess(x))
```

After that, export the preprocessed dataset for prediction purposes using machine learning.

In [18]:
```python
df1.to_csv("dataset/cleaned_tweets.csv")
df4.to_csv("dataset/cleaned_tweet_capres_2024.csv")
```

3.2. Balancing Data

The balancing process is carried out to balance the frequency of hate speech and non-hate speech into a 50:50 ratio. The balancing process begins by looking at the number of datasets that are labeled as hate speech.

In [42]:
```python
df1 = df.drop(['Tweet'], axis=1)
df1.sum()
```

Out[42]:
```
HS    5561
dtype: int64
```

There are 5561 data that are indicated as hate speech. To balance the ratio, 5561 tweet data that are labeled as non-hate speech will be taken.

In [6]:
```python
df = pd.read_csv("dataset/cleaned_tweets.csv")
df.drop("Unnamed: 0",axis=1, inplace=True)
df.head()
```

Out[6]:

	Tweet	HS
0	cowok usaha lacak perhati gue lantas remeh per...	1
1	telat tau edan sarap gue gaul cigax jifla cal ...	0
2	41 kadang pikir percaya tuhan jatuh kali kali ...	0
3	ku tau mata sipit lihat	0
4	kaum cebong kapir lihat dongok dungu haha	1

In [63]:
```python
df_HS_1 = df[df['HS'] == 1]
df_HS_0 = df[df['HS'] == 0].iloc[:5561]
df_HS_bal = pd.concat([df_HS_1, df_HS_0], axis=0)
df_HS_bal.shape
```

Out[63]:
```
(11122, 2)
```

After the balancing process is carried out, export the dataset for prediction purposes.

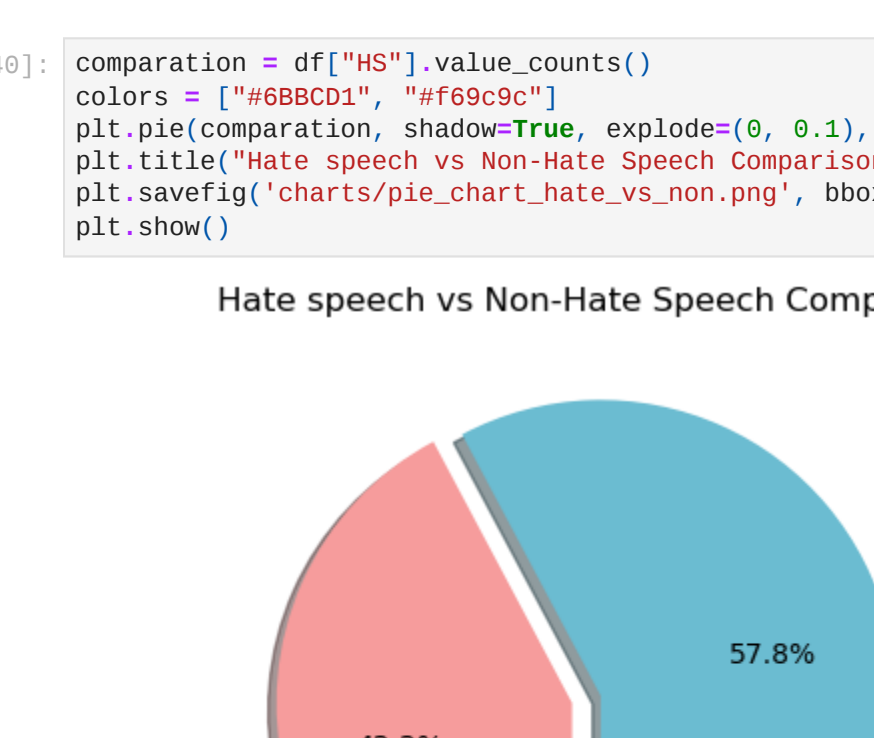
In [64]:
```python
df_HS_bal.to_csv("dataset/HS.csv")
```

4. Exploratory Data Analysis and Visualization

4.1. Hate speech vs Non-Hate speech Comparison

Comparing the number of hate speech and non-hate speech from the dataset using a pie chart.

In [46]:
```python
comparation = df["HS"].value_counts()
colors = ["#6BBCD1", "#f69c9c"]
plt.pie(comparation, shadow=True, explode=(0, 0.1), autopct='%1.1f%%', labels=["Non-Hate speech", "Hate speech"], colors=colors, startangle=270)
plt.title("Hate speech vs Non-Hate Speech Comparison")
plt.savefig('charts/pie_chart_hate_vs_non.png', bbox_inches='tight')
plt.show()
```



From the visualization provided, most of the data is non-hate speech, around 57.8%, the rest is hate speech.

4.2. The Most Frequently Used Words for Hate Speech

Visualization uses wordcloud to see what words appear most often in tweets that are indicated as hate speech.

In [41]:
```python
hs = df.loc[df["HS"]==1]
text = ' '.join(hs['Tweet'].astype(str))
whs = WordCloud(width=1000, height=800, background_color="white").generate(text)
plt.imshow(whs, interpolation="bilinear")
plt.axis("off")
plt.savefig('charts/hate_speech_wordcloud.png', bbox_inches='tight')
plt.show()
```

From the visualization, the words that appear most often are Jokowi (name of president), cebong, ganti presiden, and so on.

4.3. The Most Frequently Used Words for Non-hate Speech

Visualization uses wordcloud to see what words appear most often in tweets that are indicated as non-hate speech.

In [42]:
```python
nhs = df.loc[df["HS"]==0]
text = ' '.join(nhs['Tweet'].fillna('').astype(str))
whs = WordCloud(width=1000, height=800, background_color="white").generate(text)
plt.imshow(whs, interpolation="bilinear")
plt.axis("off")
plt.savefig('charts/non_hate_speech_wordcloud.png', bbox_inches='tight')
plt.show()
```

From the visualization, the words that appear most often are orang, gue, Indonesia, and so on.

4.4. The Most Frequently Used Words for Tweets After the Second Presidential Debate in 2024

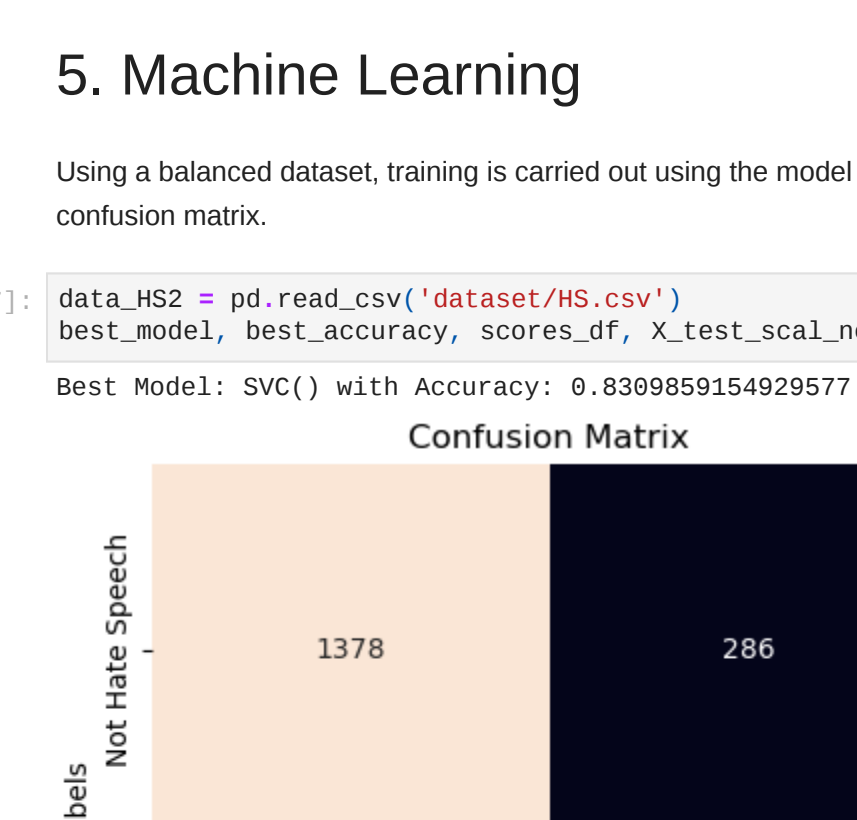
Visualization uses wordcloud to see what words appear most often in popular tweets after the second presidential debate.

In [18]:
```python
df_debat = pd.read_csv("dataset/cleaned_tweet_capres_2024.csv")
df_debat.drop("Unnamed: 0",axis=1, inplace=True)
df_debat.head()
```

Out[18]:

	Tweet
0	saudara saudara singgung singgung tanah tanah ...
1	allhamdulilah jabar 80 prabowo baliho doang ma...
2	ahahahahahahhahaha teman kelas gue pro prabowo...
3	istirahat news sumber data anies debat calon p...
4	orang pimpin rendah pijak orang puji bullyan l...

In [43]:
```python
text = ' '.join(df_debat['Tweet'].fillna('').astype(str))
whs = WordCloud(width=1000, height=800, background_color="white").generate(text)
plt.imshow(whs, interpolation="bilinear")
plt.axis("off")
plt.savefig('charts/debat_wordcloud.png', bbox_inches='tight')
plt.show()
```



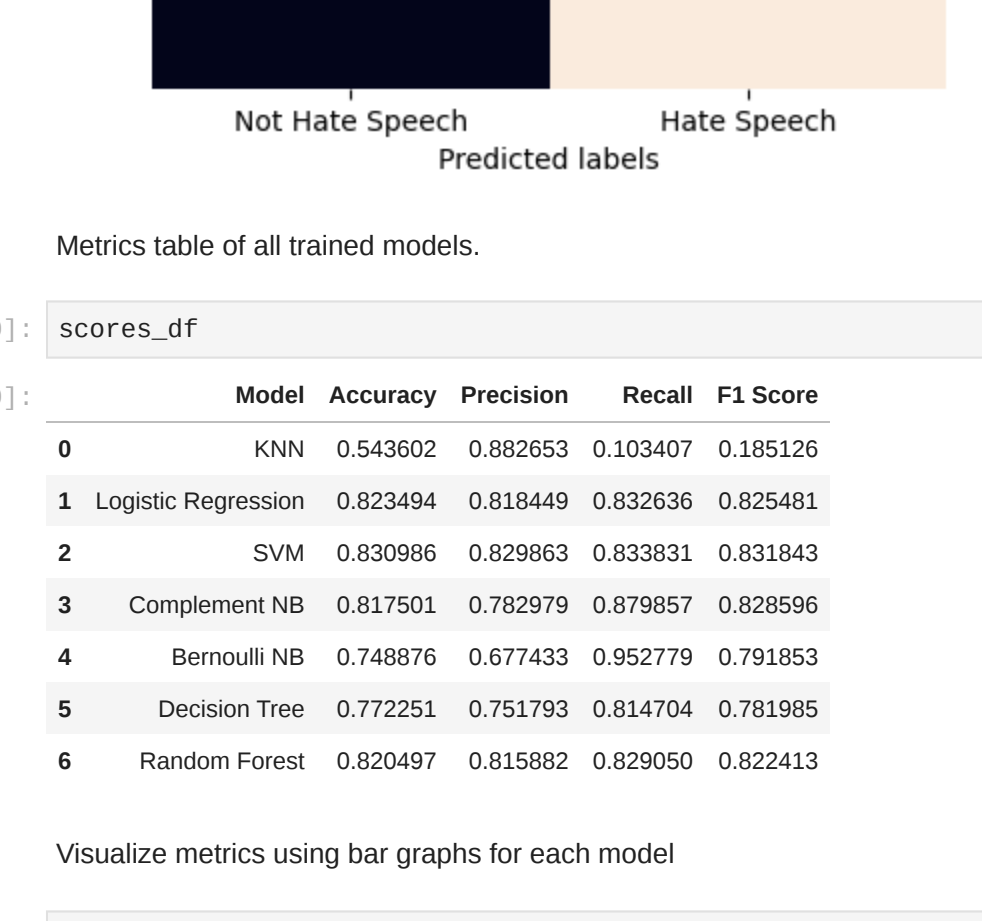
From the visualization, the words that appear most often are Prabowo and Anies (presidential candidate), Jokowi, and so on.

5. Machine Learning

Using a balanced dataset, training is carried out using the model training function. The training results show that the best model is SVM with an accuracy of 83% with prediction results represented using a confusion matrix.

In [47]:
```python
data_HS2 = pd.read_csv('dataset/HS.csv')
best_model, best_accuracy, scores_df, X_test_scal_new = vector_model(data_HS2, 'HS', (1,2))
```

```
Best Model: SVC() with Accuracy: 0.8309985915492977
```



Metrics table of all trained models.

In [20]:
```python
scores_df
```

Out[20]:

	Model	Accuracy	Precision	Recall	F1 Score
0	KNN	0.543602	0.882653	0.103407	0.185126
1	Logistic Regression	0.823494	0.818449	0.832636	0.825481
2	SVM	0.830986	0.829863	0.833831	0.831843
3	Complement NB	0.817501	0.782979	0.879857	0.828596
4	Bernoulli NB	0.748876	0.677433	0.952779	0.791853
5	Decision Tree	0.772251	0.751793	0.814704	0.781985
6	Random Forest	0.820497	0.815882	0.829050	0.822413

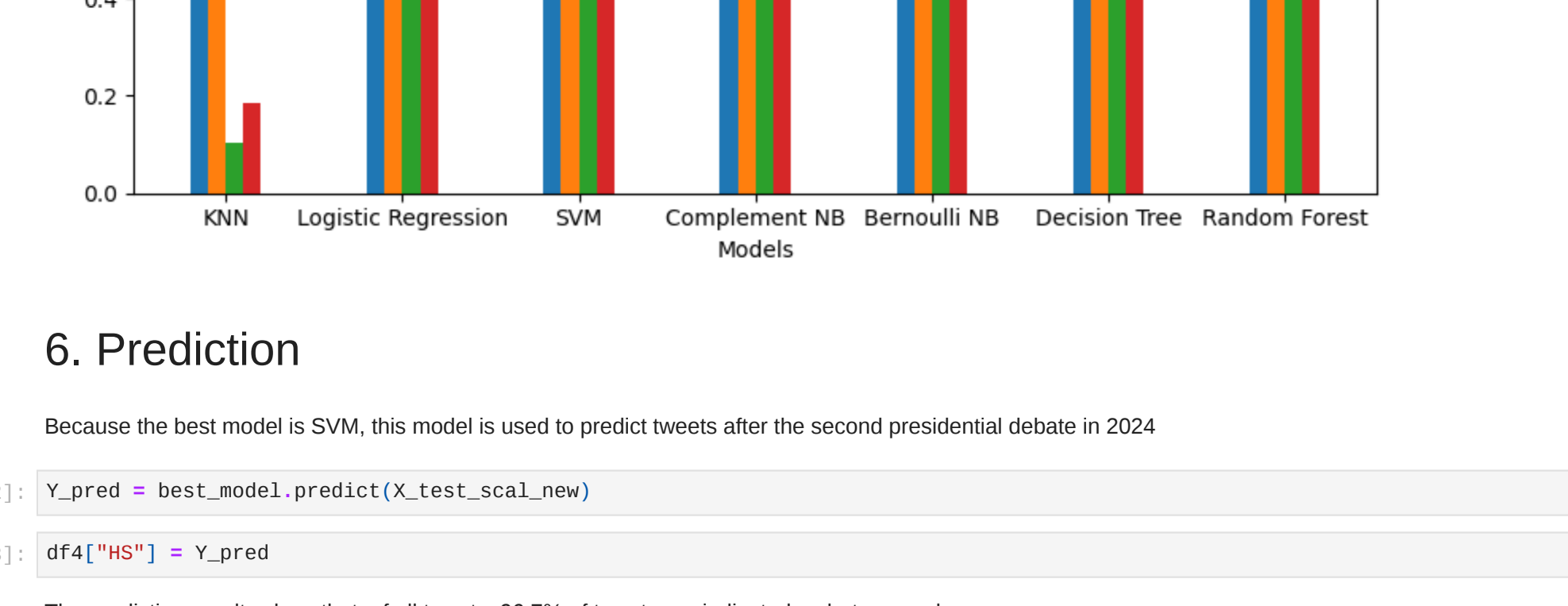
Visualize metrics using bar graphs for each model

In [46]:
```python
viz_scores = scores_df.copy()
viz_scores.set_index('Model', inplace=True)

plt.figure(figsize=(10, 4))

metrics = viz_scores.columns.tolist()
for i, metric in enumerate(metrics):
    values = viz_scores[metric].values
    x = range(len(viz_scores))
    plt.bar(x=[pos + i * 0.1 for pos in x], height=values, width=0.1, label=metric)

plt.xlabel('Models')
plt.ylabel('Values')
plt.title('Metrics for Different Models')
plt.xticks([i + 0.15 for i in x], viz_scores.index)
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left')
plt.tight_layout()
plt.savefig('charts/metrics_bar_chart.png', bbox_inches='tight')
plt.show()
```



6. Prediction

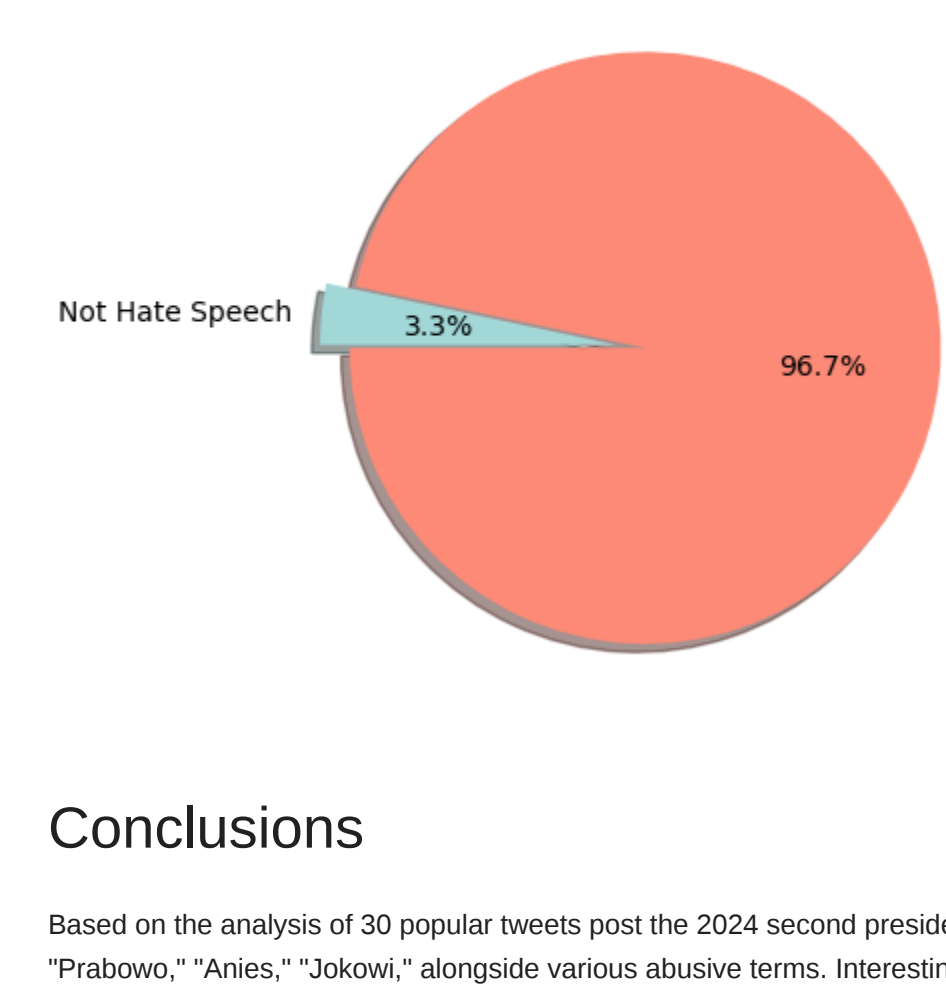
Because the best model is SVM, this model is used to predict tweets after the second presidential debate in 2024

In [22]:
```python
Y_pred = best_model.predict(X_test_scal_new)
```

In [23]:
```python
df4["HS"] = Y_pred
```

The prediction results show that, of all tweets, 96.7% of tweets are indicated as hate speech

In [45]:
```python
comparation = df4["HS"].value_counts()
colors = ["#fc8a77", "#a2d7d8"]
plt.pie(comparation, shadow=True, explode=(0, 0.1), autopct='%1.1f%%', labels=["Hate Speech", "Not Hate Speech"], colors=colors,startangle=180)
plt.title("Hate Speech vs Not Hate Speech Comparison")
plt.savefig('charts/pie_chart_debat_comp.png', bbox_inches='tight')
plt.show()
```



Conclusions

Based on the analysis of 30 popular tweets post the 2024 second presidential debate, a staggering 96.7% of these tweets were identified as hate speech. The recurrently used words in these tweets included "Prabowo," "Anies," "Jokowi," alongside various abusive terms. Interestingly, these align closely with the frequently occurring terms observed in the training dataset from 2018, featuring "Jokowi" and "Prabowo" along with similar abusive language. This suggests a persistent usage of similar word patterns over the years.

To enhance future analysis, augmenting the training dataset with data from both the first and second presidential debates could offer new tweet patterns. Additionally, employing an neural-network model such as LSTM instead of a conventional machine learning model might enhance the ability to discern patterns within tweets, potentially elevating prediction accuracy.

References

- Komnas HAM. 2015. *Buku Saku Penanganan Ujaran Kebencian (Hate Speech)*. Komisi Nasional Hak Asasi Manusia, Jakarta.
- Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. *Automated hate speech detection and the problem of offensive language. In International AAAI Conference on Web and Social Media (ICWSM)*, pages 512–515.
- I Dewa Putu Wijana and Muhammad Rohmadi. 2010. *Sosiolinguistik: Kajian, Teori, dan Analisis*. Pustaka Pelajar, Yogyakarta.
- Ika Alfina, Rio Mulia, Mohamad Ivan Fanany, and Yudo Ekanata. 2017. Hate speech detection in the indonesian language: A dataset and preliminary study. *In International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 233–238.
- Tansa Trisna Astono Putri. 2018. Analisis dan deteksi hate speech pada sosial twitter berbahasa indonesia. Master's thesis, Faculty of Computer Science, Universitas Indonesia.
- Muhammad Okky Ibrohim and Indra Budi. 2019. Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter.
- Tumasjan, A., Andranik, S., Sprenger, T., Sandner, P., & Welpe, I. (2010). Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. Word: Journal of the International Linguistic Association, 10.
- Kurniawan, S., Gata, W., Puspitawati, D. A., -, N., Tabrani, M., & Novel, K. (2019). Perbandingan Metode Klasifikasi Analisis Sentimen Tokoh Politik Pada Komentar Media Berita Online. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 3(2), 176 - 183. https://doi.org/10.29207/resti.v3i2.935
- Ansari, M. Z., Aziz, M. B., Siddiqui, M. O., Mehra, H., & Singh, K. P. (2020). Analysis of Political Sentiment Orientations on Twitter. Procedia Computer Science, 167, 1821-1828. https://doi.org/10.1016/j.procs.2020.03.201
- Omran, E., Al Tararwah, E., & Al Qundus, J. (2023). A comparative analysis of machine learning algorithms for hate speech detection in social media. Online Journal of Communication and Media Technologies, 13(4), e202348. https://doi.org/10.30935/ojcmt/13603