Analiza Sentymentów Opinie Hotelowych za pomocą Modeli Uczenia Maszynowego

Projekt analizy sentymentów na podstawie opinii hotelowych

Łukasz Syguła
15.08.2024

Cel Projektu

- Zbadanie i ocena różnych modeli uczenia maszynowego dla analizy sentymentów w opiniach hotelowych.

- Identyfikacja najlepszego modelu do przewidywania sentymentu na podstawie opinii klientów.

Zakres Projektu

Projekt obejmował następujące etapy:

- **Zbieranie i przygotowanie danych:** Zebrałem dane dotyczące opinii hotelowych, oczyściłem je i przygotowałem do analizy.
- **Rozwój modeli:** Zaimplementowałem różne modele klasyfikacyjne, takie jak Naive Bayes, Support Vector Machines (SVM), Regresja Logistyczna, Random Forest i K-Nearest Neighbors (KNN).
- **Ewaluacja modeli:** Każdy model został oceniony pod kątem dokładności, precyzji, czułości i wskaźnika F1.
- **Optymalizacja:** Przeprowadziłem tuning hiperparametrów za pomocą GridSearchCV, aby poprawić wyniki modeli.
- **Prezentacja wyników:** Wyniki zostały zaprezentowane za pomocą metryk i wizualizacji.

Data Cleaning and Preprocessing

In [7]:
```
1 df.isnull().sum()
```

Out[7]:
```
ratings                0
title                  0
text                   0
author                 0
date_stayed          658
offering_id            0
num_helpful_votes      0
date                   0
id                     0
via_mobile             0
dtype: int64
```

In [8]:
```
1 df.drop(columns=['id'], inplace=True)
```

In [9]:
```
1 df.dropna(subset=['date_stayed'], inplace=True)
```

In [10]:
```
1 df.head()
```

Out[10]:

	ratings	title	text	author	date_stayed	offering_id	num_helpful_votes	date	via_mobile
340013	{'service': 5.0, 'cleanliness': 5.0, 'overall'...	“Very nice experience for a country boy going ...	Being from a small town in Tennessee, I was ve...	{'username': 'Tucker124', 'num_reviews': 1, 'i...	October 2010	111492	2	2010-10-25	False

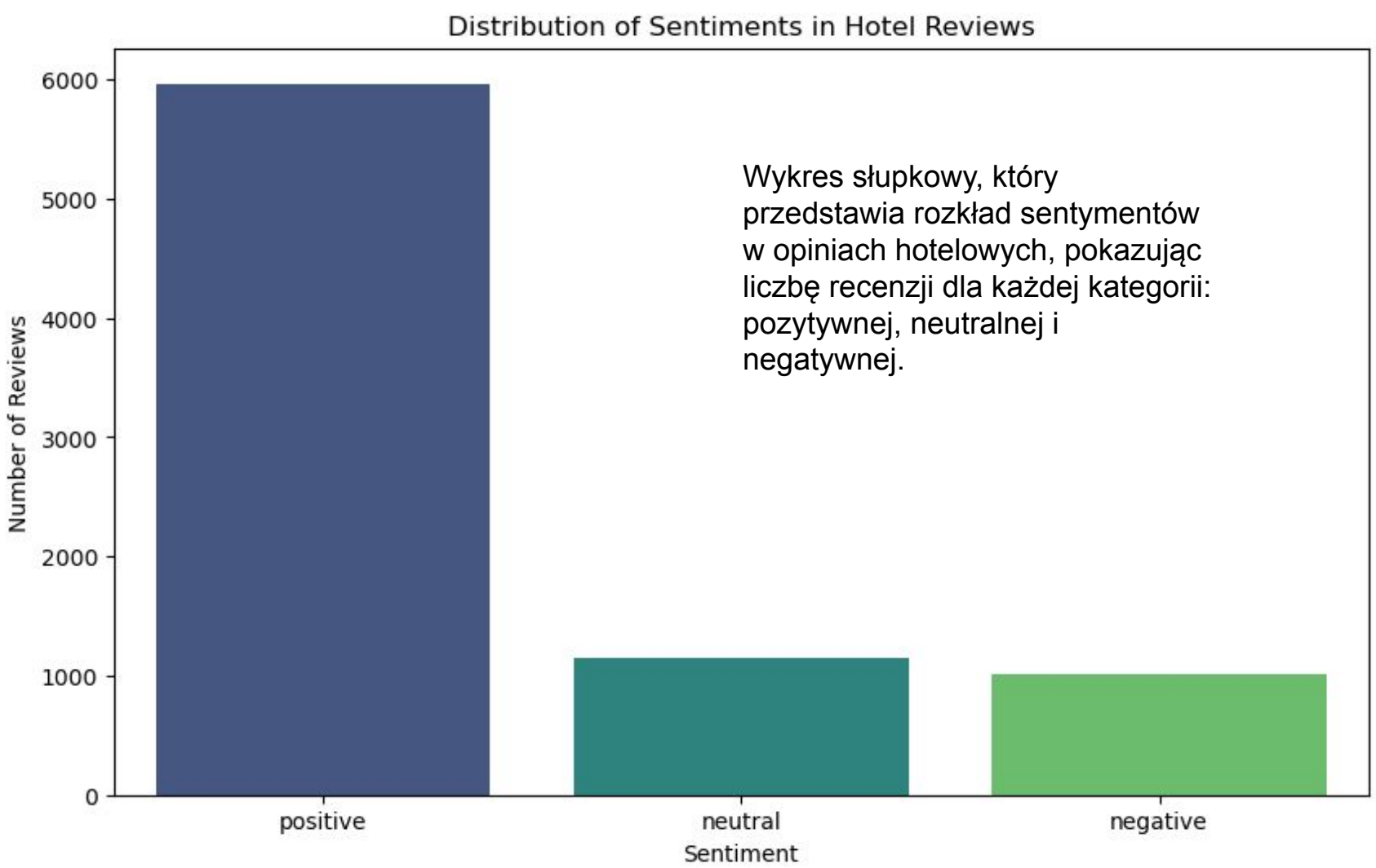
Classifies sentiment based on the 'overall' rating in the dictionary

In [13]:
```
def get_sentiment(ratings):
    overall_rating = ratings.get('overall')
    if overall_rating >= 4:
        return 'positive'
    elif overall_rating == 3:
        return 'neutral'
    else:
        return 'negative'
```

In [14]:
```
df['sentiment'] = df['ratings'].apply(eval).apply(get_sentiment)
```

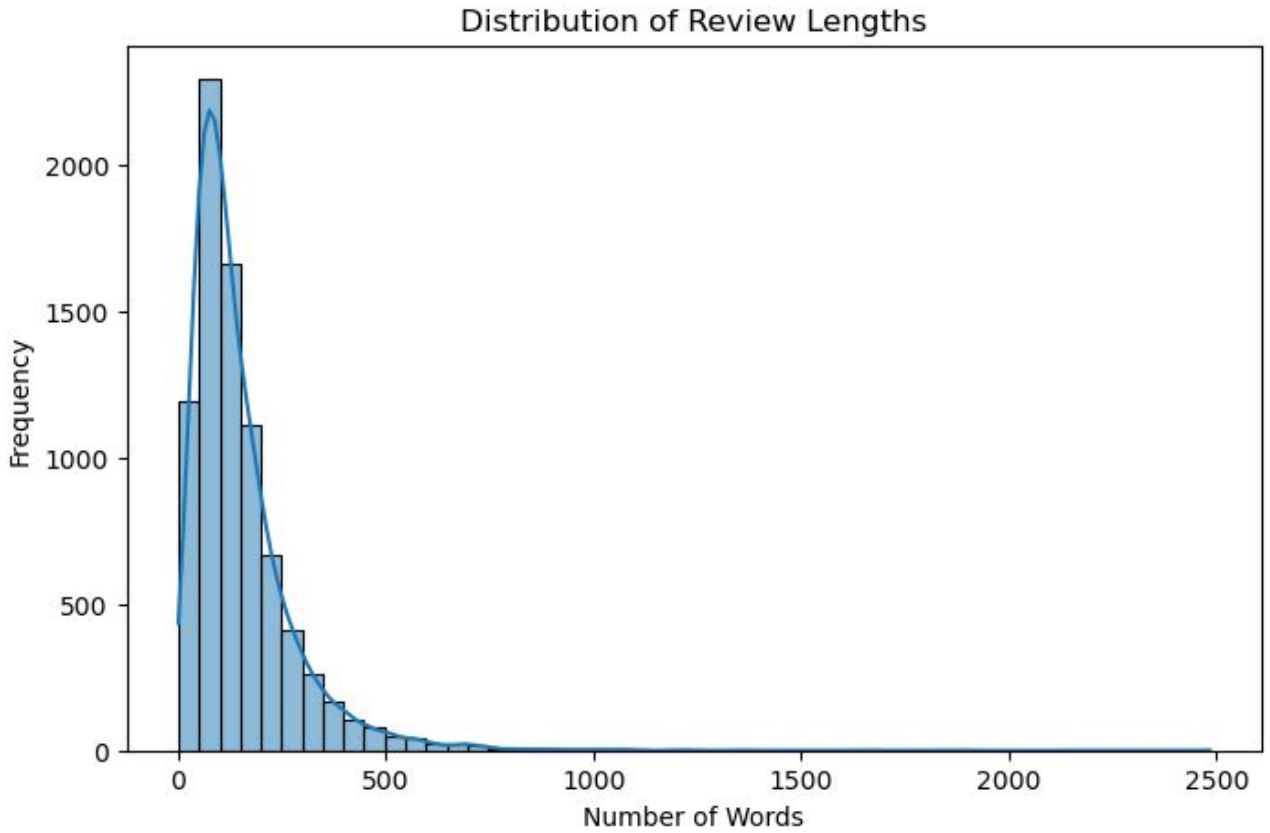
In [15]:
```
sentiment_counts = df['sentiment'].value_counts()
```

In [16]:
```
plt.figure(figsize=(10, 6))

sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values, palette='viridis')

plt.title('Distribution of Sentiments in Hotel Reviews')
plt.xlabel('Sentiment')
plt.ylabel('Number of Reviews')

plt.show()
```

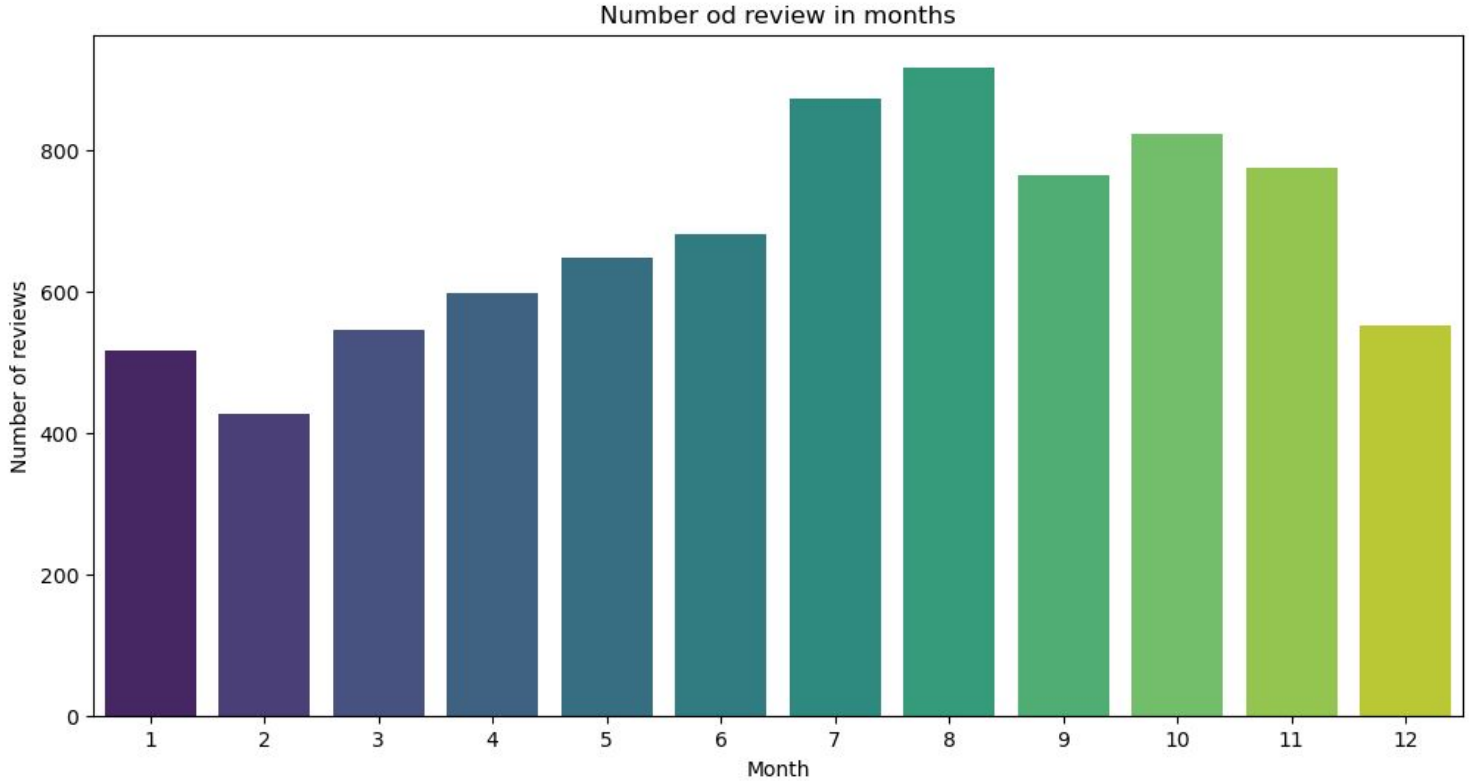


Wykres słupkowy, który przedstawia rozkład sentymentów w opiniach hotelowych, pokazując liczbę recenzji dla każdej kategorii: pozytywnej, neutralnej i negatywnej.

Histogram, który pokazuje rozkład długości opinii hotelowych w liczbie słów, ilustrując, jak zróżnicowana jest długość recenzji w zbiorze danych.

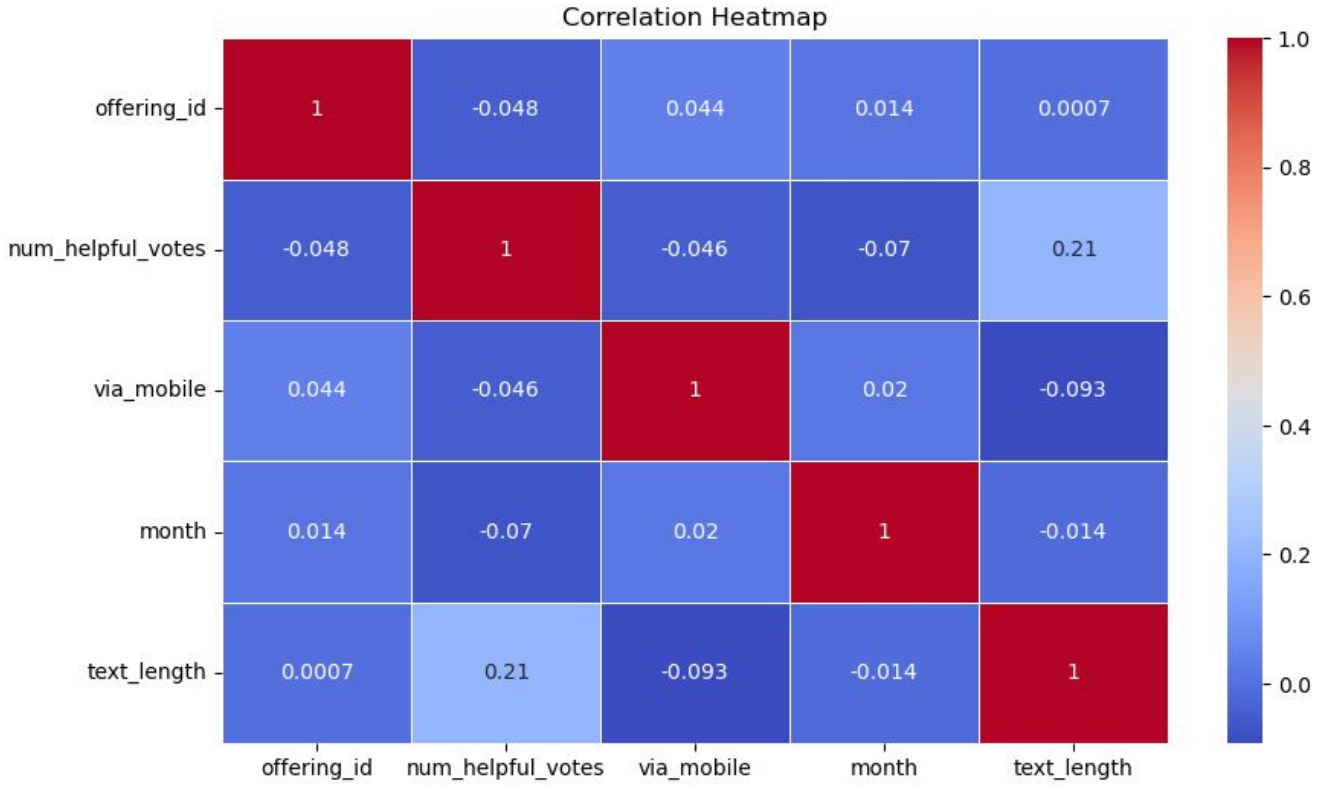


Wykres słupkowy przedstawiający liczbę recenzji hotelowych w poszczególnych miesiącach, co pozwala zobaczyć, jak często opinie były zgłaszane w różnych okresach roku.



Wizualizacja w postaci chmury słów, która przedstawia najczęściej używane słowa we wszystkich recenzjach hotelowych.

Na tym slajdzie dodałem mapę cieplną korelacji, która pokazuje zależności pomiędzy różnymi cechami numerycznymi w zbiorze danych.



Metodologia:

Wykorzystałem podejście polegające na:

- **Czyszczeniu danych:** Usunięciu brakujących wartości i nieistotnych zmiennych oraz konwersji danych do odpowiednich formatów.
- **Przetwarzaniu tekstu:** Zastosowałem lematyzację i usuwanie stopwords, a następnie przekształciłem teksty opinii w cechy numeryczne za pomocą TF-IDF.
- **Budowaniu modeli:** Wykorzystałem modele takie jak Naive Bayes, SVM, Regresja Logistyczna, Random Forest i KNN do klasyfikacji sentymentu.

Feature Extraction

In [26]:
```
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)
```

In [27]:
```
df['cleaned_text'] = df['text'].apply(preprocess_text)
```

In [28]:
```
df[['text', 'cleaned_text']].head()
```

Out[28]:

	text	cleaned_text
340013	Being from a small town in Tennessee, I was ve...	small town tennessee unsure expect large city ...
477333	I stayed at this courtyard for 2 nights . Ever...	stayed courtyard night everything great staff ...
755575	Even in Boston for $180 plus taxes per night y...	even boston plus tax per night might expect be...
709674	This hotel is a great old building (formerly t...	hotel great old building formerly paso del nor...
799143	The Good~room was larger than expected, free i...	goodroom larger expected free internet room ba...

In [29]:
```
tfidf = TfidfVectorizer(max_features=1000)
```

In [30]:
```
X = tfidf.fit_transform(df['cleaned_text']).toarray()
y = df['sentiment']
```

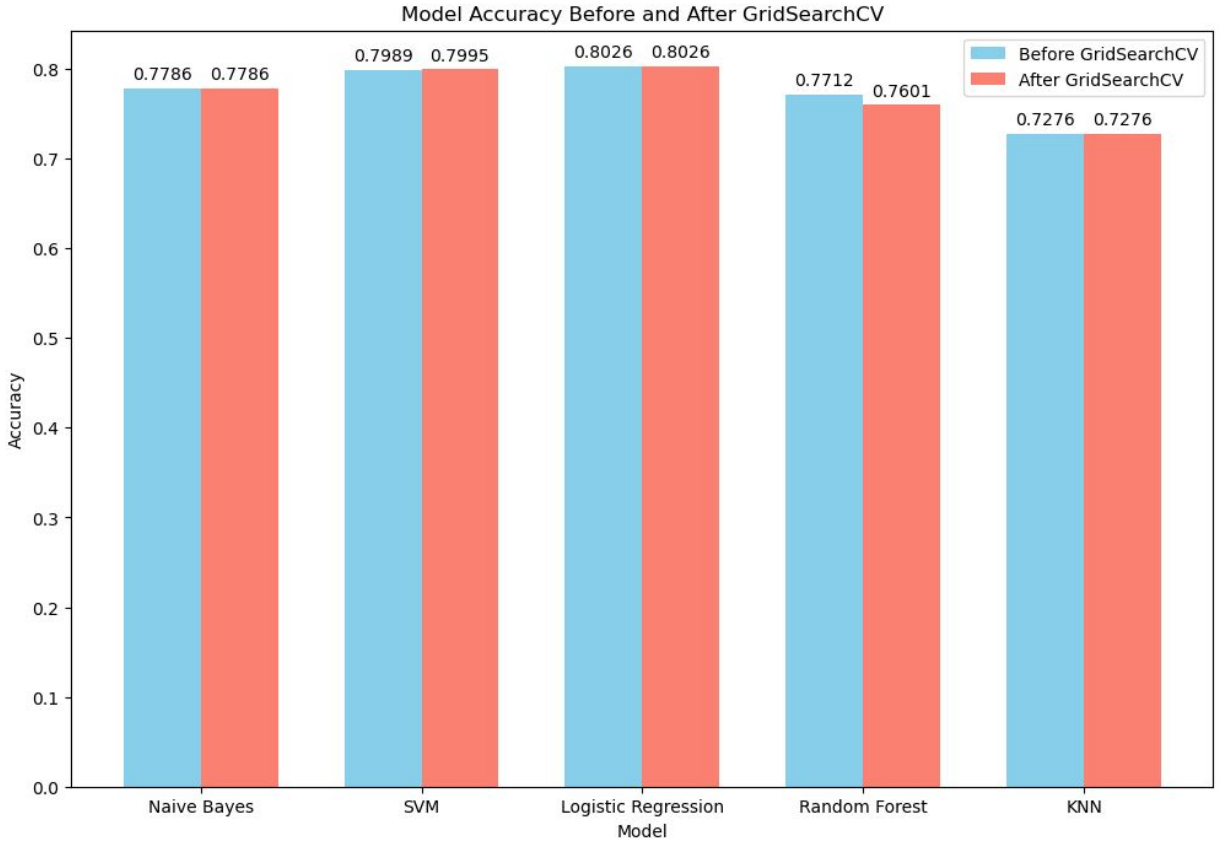
In [31]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Wyzwania i rozwiązania:

Podczas realizacji projektu napotkałem na kilka wyzwań:

- **Niskie wyniki modeli na klasie neutralnej:** Modele, takie jak Naive Bayes i KNN, miały trudności z klasyfikacją opinii neutralnych. Rozwiązaniem było testowanie bardziej złożonych modeli, takich jak SVM i Regresja Logistyczna.
- **Optymalizacja modeli:** Po zastosowaniu GridSearchCV wyniki niektórych modeli, jak Random Forest, nieznacznie się pogorszyły. Może to wynikać z nadmiernego dopasowania modelu do danych treningowych.

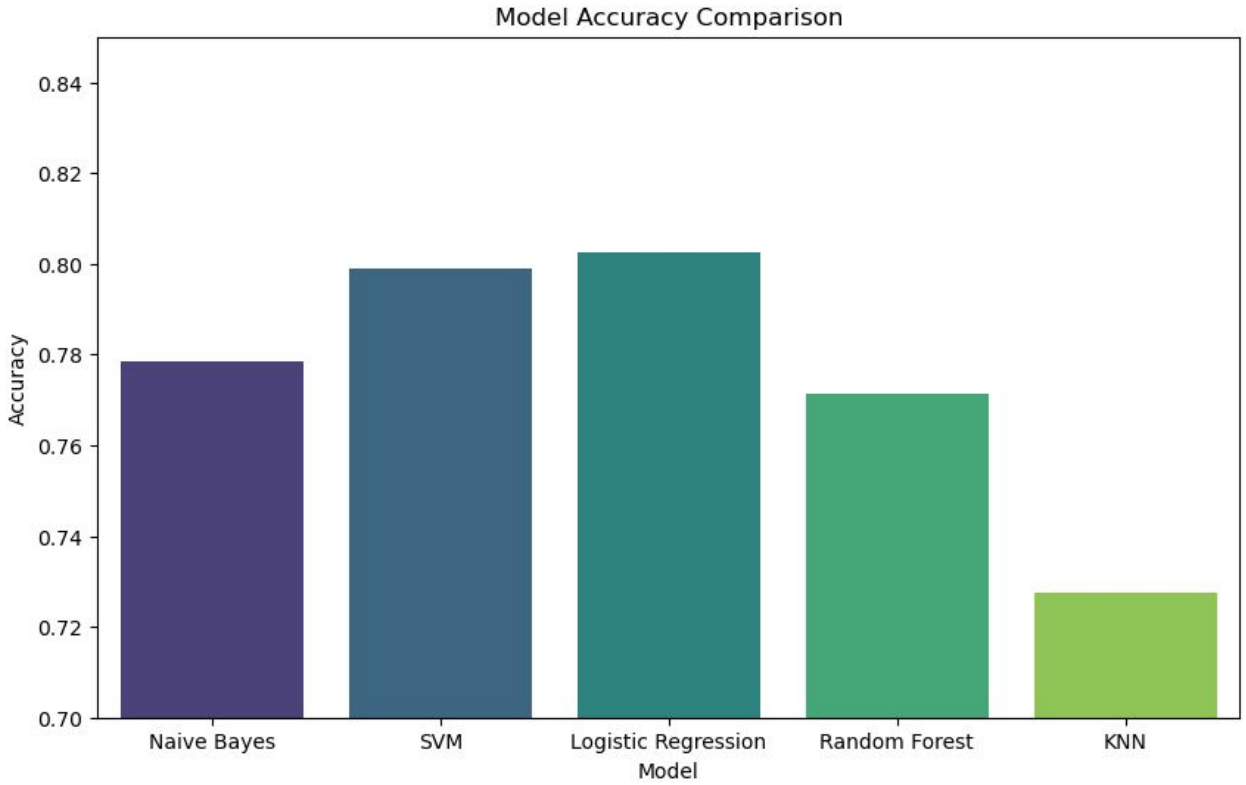
Na tym slajdzie umieściłem wykres słupkowy porównujący dokładność modeli przed i po zastosowaniu GridSearchCV. Wykres ilustruje, jak tuning hiperparametrów wpłynął na wyniki modeli, pokazując zarówno pozytywne, jak i negatywne zmiany w dokładności.



Wyniki:

- **Najlepszy model:** Regresja Logistyczna osiągnęła najwyższą dokładność na poziomie 80.26%, co czyni ją najlepszym modelem w projekcie.
- **Porównanie modeli:** SVM również uzyskał solidne wyniki, ale ustąpił Regresji Logistycznej. Random Forest, mimo optymalizacji, osiągnął gorsze wyniki. Naive Bayes i KNN okazały się najmniej efektywne, zwłaszcza w klasyfikacji opinii neutralnych.

wykres słupkowy, który pokazuje porównanie dokładności różnych modeli użytych w projekcie. Wykres podkreśla, który model osiągnął najwyższą dokładność, a także różnice pomiędzy pozostałymi modelami.



Wnioski:

Projekt pokazał, że **Regresja Logistyczna** jest obecnie najskuteczniejszym modelem do analizy sentymentu opinii hotelowych w tej analizie. Optymalizacja hiperparametrów może poprawić wyniki niektórych modeli, ale nie zawsze gwarantuje poprawę. Ważnym wnioskiem jest potrzeba dalszych badań nad metodami lepszej klasyfikacji opinii neutralnych.

Refleksja:

Realizacja projektu pozwoliła mi zdobyć doświadczenie w zakresie:

- **Przetwarzania danych tekstowych i ich analizie za pomocą różnych modeli uczenia maszynowego.**
- **Zrozumienia, jak tuning hiperparametrów może wpłynąć na wydajność modeli.**
- **Analizy wyników i wyciągania wniosków dotyczących potencjalnych usprawnień.**

Gdybym miał realizować projekt ponownie, skupiłbym się na eksploracji bardziej zaawansowanych metod przetwarzania tekstu oraz innych technik klasyfikacyjnych, aby lepiej radzić sobie z opiniami neutralnymi.