```
import numpy as np
import pandas as pd
import chardet


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report


with open("/content/drive/MyDrive/Colab Notebooks/Suicide_Detection.csv", 'rb') as f:
  encoding = chardet.detect(f.read())['encoding']


data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Suicide_Detection.csv", encoding=encoding)


# Define a mapping from labels to numbers
label_mapping = {'suicide': 1, 'non-suicide': 0}

# Replace the labels in the 'label' column
data['class'] = data['class'].map(label_mapping)


data=data.dropna()
data.describe()
# data.head()
```

	Unnamed: 0	class
count	41337.000000	41337.000000
mean	31045.271573	0.497859
std	17901.162606	0.500001
min	2.000000	0.000000
25%	15534.000000	0.000000
50%	31114.000000	0.000000
75%	46572.000000	1.000000
max	61976.000000	1.000000

```
vectorizer=CountVectorizer(analyzer='word')
X=vectorizer.fit_transform(data['text'])
X
```

```
<41337x64859 sparse matrix of type '<class 'numpy.int64'>'
        with 2943354 stored elements in Compressed Sparse Row format>
```

```
X_train, X_test, y_train, y_test = train_test_split(X, data['class'], test_size=0.2, random_state=1)


model=LogisticRegression(max_iter=10000)


model.fit(X_train,y_train)
```

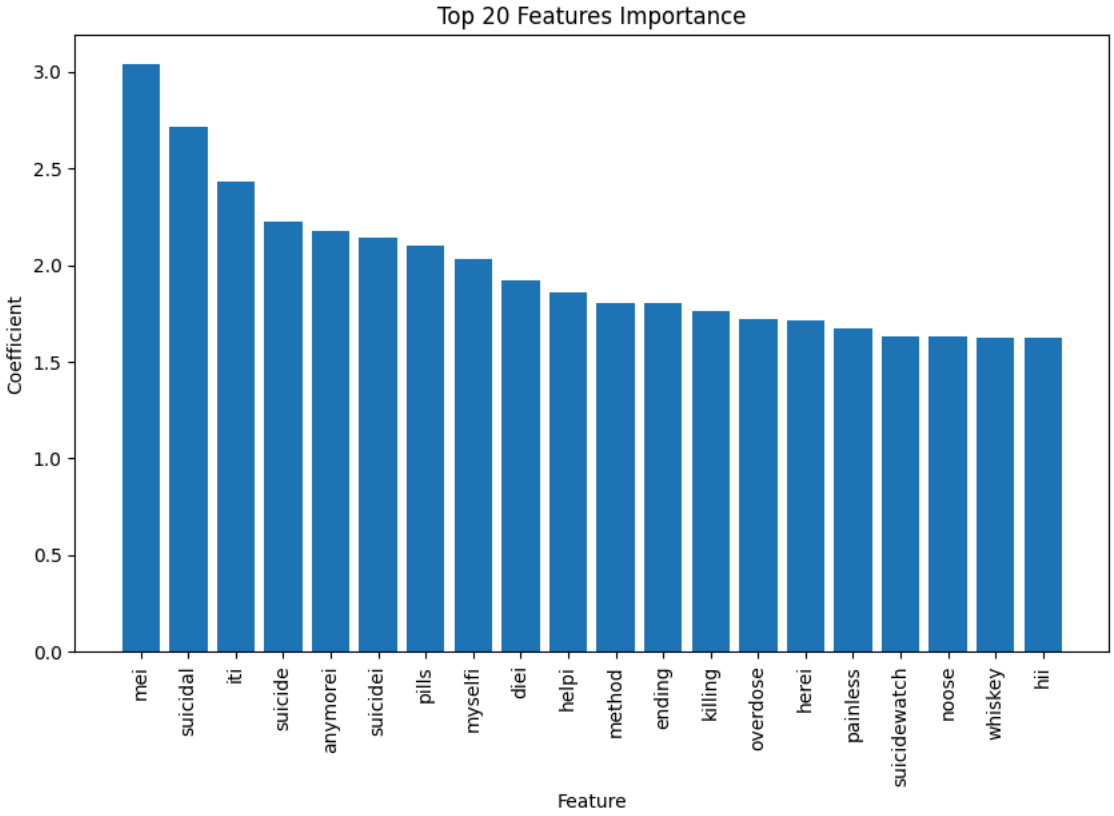
```
LogisticRegression(max_iter=10000)
```

Generate visualize the model Close

1 of 4 Use code with caution

```
# prompt: visualize the model

import matplotlib.pyplot as plt

# Get the feature names from the vectorizer
feature_names = vectorizer.get_feature_names_out()

# Get the coefficients from the model
coefficients = model.coef_.tolist()[0]

# Create a DataFrame with the feature names and coefficients
df_coef = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

# Sort the DataFrame by the coefficients
```

```
# Sort the DataFrame by the coefficients
df_coef = df_coef.sort_values(by='Coefficient', ascending=False)

# Plot the top 20 features
plt.figure(figsize=(10, 6))
plt.bar(df_coef['Feature'][:20], df_coef['Coefficient'][:20])
plt.xticks(rotation=90)
plt.xlabel('Feature')
plt.ylabel('Coefficient')
plt.title('Top 20 Features Importance')
plt.show()
```




```
# Evaluate the model
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
    --NORMAL--
```

	precision	recall	f1-score	support
0.0	0.90	0.95	0.92	4204
1.0	0.94	0.90	0.92	4064
accuracy			0.92	8268
macro avg	0.92	0.92	0.92	8268
weighted avg	0.92	0.92	0.92	8268