```
import os
import pandas as pd
import numpy as np
```

```
movie_df = pd.read_csv(f"C:\\Users\\DELL\\Downloads\\IMDB MOVIE\\IMDB DATASET.csv")
```

```
movie_df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
movie_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

```
X = movie_df['review']
y = movie_df['sentiment']
```

```
count = CountVectorizer()
Vectorizer = TfidfVectorizer(stop_words='english', max_features=100)
X = Vectorizer.fit_transform(movie_df['review'])
tfidf_score = dict(zip(Vectorizer.get_feature_names_out(), X.sum(axis=0).A1))
```

```
print(tfidf_score)
```

```
{'10': np.float64(1416.2310009191463), 'acting': np.float64(1915.6953236796444), 'action': np.float64(1136.8769146695), 'actors': np.float64(1468.7403940129602),
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)
```

```
log = LogisticRegression(max_iter=1000)
log.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

```
y_pred = log.predict(X_test)
```

```
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

    negative       0.72      0.73      0.72      4926
    positive       0.73      0.73      0.73      5074

    accuracy                           0.73     10000
   macro avg       0.73      0.73      0.73     10000
weighted avg       0.73      0.73      0.73     10000
```

```
print(confusion_matrix(y_pred, y_test))
```

```
[[3580 1346]
 [1381 3693]]
```

```
from wordcloud import WordCloud
wordcloud = WordCloud(width=500, height=300, background_color='black', colormap='plasma')
wordcloud = wordcloud.generate_from_frequencies(tfidf_score)
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Top Tfidf Words")
plt.show()
```