```
import pandas as pd
import re

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

```
df = pd.read_csv("BTC_tweets_daily_example.csv")
print(df.head())
```

```
  Unnamed: 0                         Date  \
0          0  Fri Mar 23 00:40:32 +0000 2018
1          1  Fri Mar 23 00:40:34 +0000 2018
2          2  Fri Mar 23 00:40:35 +0000 2018
3          3  Fri Mar 23 00:40:36 +0000 2018
4          4  Fri Mar 23 00:40:36 +0000 2018

                                               Tweet     Screen_name  \
0  RT @ALXTOKEN: Paul Krugman, Nobel Luddite. I h...  myresumerocket
1  @lopp @_Kevin_Pham @psycho_sage @naval But @Pr...        BitMocro
2  RT @tippereconomy: Another use case for #block...    hojachotopur
3                free coins https://t.co/DiuoePJdap   denies_distro
4  RT @payvxofficial: WE are happy to announce th...      aditzgraha

                                     Source  \
0                                        []
1                                [u'Bitcoin']
2  [u'blockchain', u'Tipper', u'TipperEconomy']
3                                        []
4                                        []

                                                Link     Sentiment  \
0  <a href="http://twitter.com" rel="nofollow">Tw...   ['neutral']
1  <a href="http://twitter.com/download/android" ...   ['neutral']
2  <a href="http://twitter.com" rel="nofollow">Tw...  ['positive']
3  <a href="http://twitter.com" rel="nofollow">Tw...  ['positive']
4  <a href="http://twitter.com/download/android" ...  ['positive']

   sent_score  New_Sentiment_Score  New_Sentiment_State
0         0.0             0.000000                  0.0
1         0.0             0.000000                  0.0
2         1.0             0.136364                  1.0
3         1.0             0.400000                  1.0
4         1.0             0.468182                  1.0
```

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"#\w+", "", text)
    text = re.sub(r"[^a-z\s]", "", text)
    return text

```

```
# Step 4: Apply cleaning
df["Tweet"] = df["Tweet"].astype(str)
df["Tweet"] = df["Tweet"].apply(clean_text)
```

```
# Step 5: Split data (IMPORTANT – avoids leakage)
# Remove rows where 'New_Sentiment_State' is NaN
df_cleaned = df.dropna(subset=['New_Sentiment_State'])

X = df_cleaned["Tweet"]
y = df_cleaned["New_Sentiment_State"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

```

```
# Step 6: TF-IDF Vectorization
tfidf = TfidfVectorizer(stop_words="english", max_features=5000)

X_train_tfidf = tfidf.fit_transform(X_train)   # fit ONLY on training data
X_test_tfidf = tfidf.transform(X_test)

```

```
# Step 7: Train Logistic Regression model
model = LogisticRegression(max_iter=1000)
```

```
3
4 model.fit(X_train_tfidf, y_train)
5
```

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

```
1 # Step 8: Predictions
2 y_pred = model.predict(X_test_tfidf)
3
```

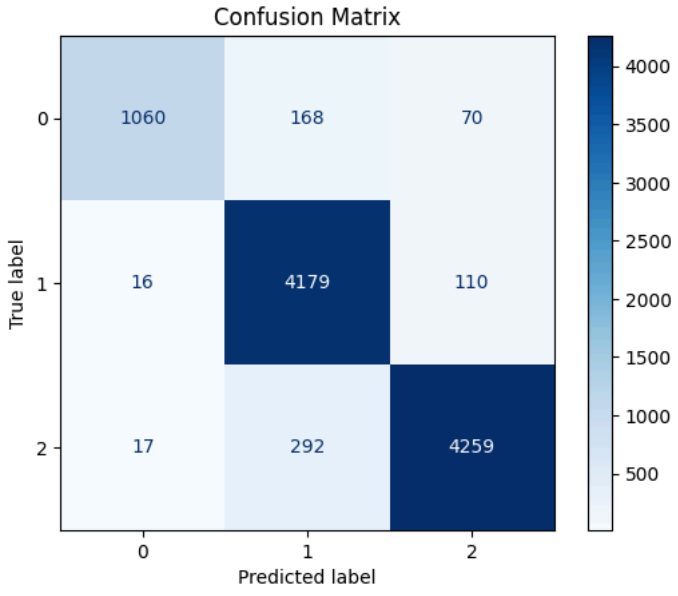
```
1 # Step 9: Evaluation
2 print("Accuracy:", accuracy_score(y_test, y_pred))
3 print("\nClassification Report:\n")
4 print(classification_report(y_test, y_pred))
5
```

```
Accuracy: 0.9338314816635532

Classification Report:

              precision    recall  f1-score   support

        -1.0       0.97      0.82      0.89      1298
         0.0       0.90      0.97      0.93      4305
         1.0       0.96      0.93      0.95      4568

    accuracy                           0.93     10171
   macro avg       0.94      0.91      0.92     10171
weighted avg       0.94      0.93      0.93     10171
```

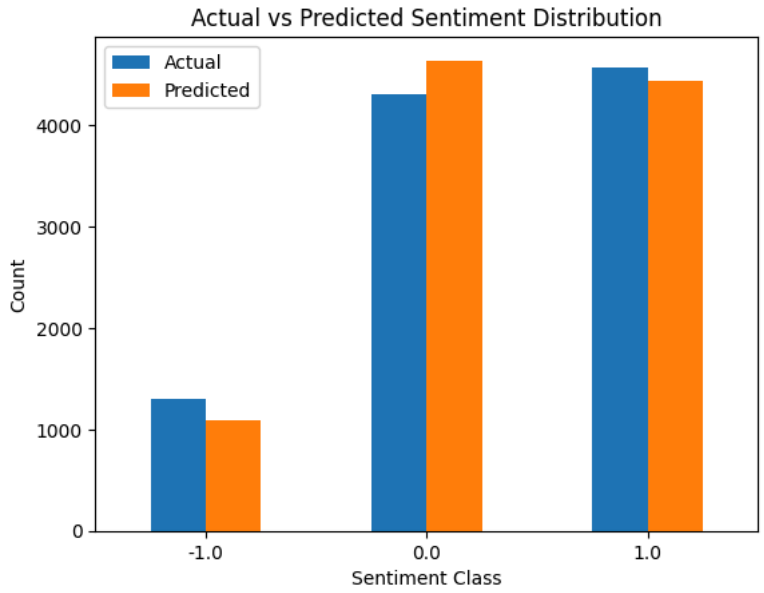
```
1 # Step 10: Predict sentiment of new BTC tweet
2 sample_tweet = ["Charan is good boy"]
3 sample_clean = [clean_text(sample_tweet[0])]
4 sample_tfidf = tfidf.transform(sample_clean)
5 prediction = model.predict(sample_tfidf)
6 probability = model.predict_proba(sample_tfidf)
7 print("Predicted Sentiment:", prediction[0])
8 print("Probabilities:", probability)
```

```
Predicted Sentiment: 1.0
Probabilities: [[0.00735989 0.02634762 0.96629249]]
```

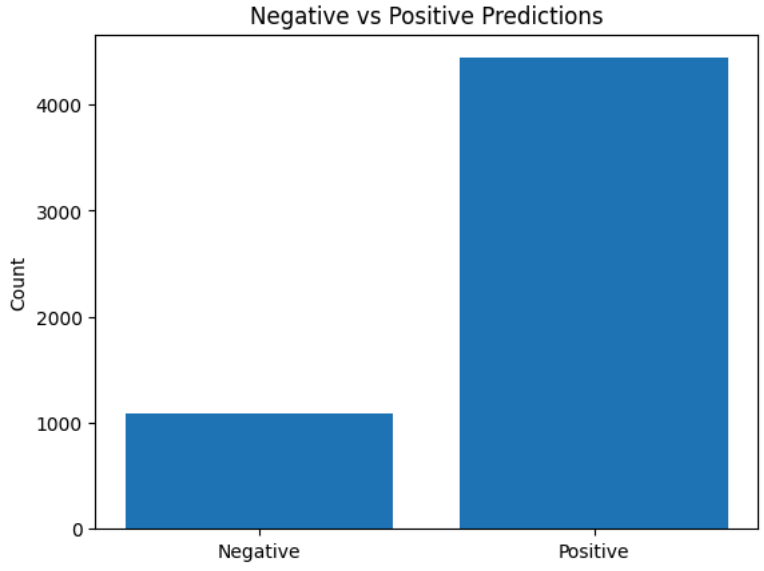
```
1 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
2 import matplotlib.pyplot as plt
3
4 cm = confusion_matrix(y_test, y_pred)
5
6 disp = ConfusionMatrixDisplay(confusion_matrix=cm)
7 disp.plot(cmap="Blues")
8 plt.title("Confusion Matrix")
9 plt.show()
10
```



```
import pandas as pd

actual_counts = pd.Series(y_test).value_counts().sort_index()
pred_counts = pd.Series(y_pred).value_counts().sort_index()

df_plot = pd.DataFrame({
    "Actual": actual_counts,
    "Predicted": pred_counts
})

df_plot.plot(kind="bar")
plt.title("Actual vs Predicted Sentiment Distribution")
plt.xlabel("Sentiment Class")
plt.ylabel("Count")
plt.xticks(rotation=0)
plt.show()

```

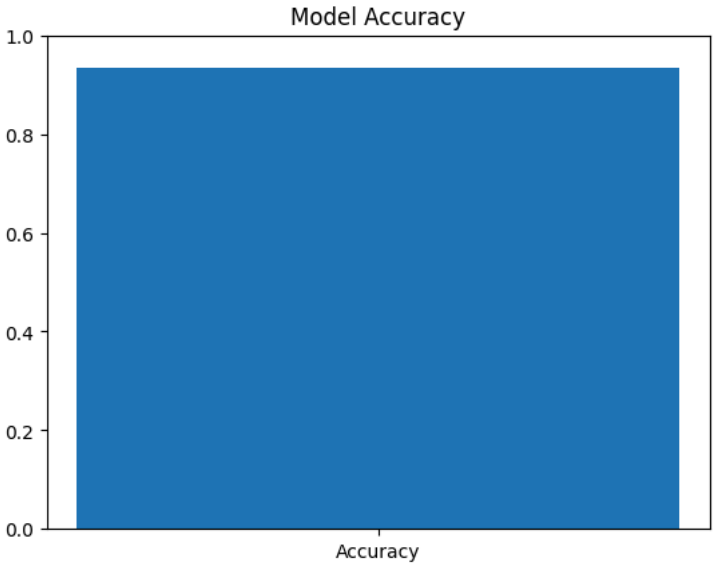



```
neg_pos = {
    "Negative": pred_counts.get(-1, 0),
    "Positive": pred_counts.get(1, 0)
}

plt.bar(neg_pos.keys(), neg_pos.values())
plt.title("Negative vs Positive Predictions")
plt.ylabel("Count")
plt.show()

```

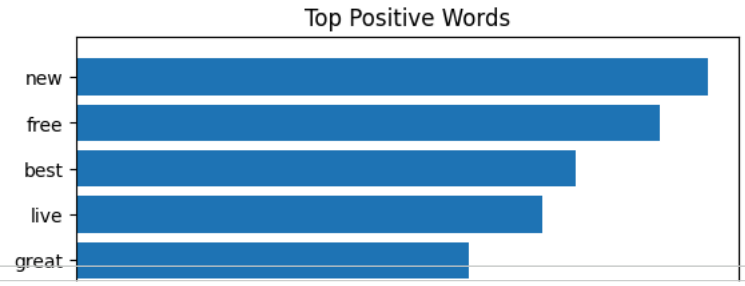



```
acc = accuracy_score(y_test, y_pred)

plt.bar(["Accuracy"], [acc])
```

```
plt.ylim(0, 1)
plt.title("Model Accuracy")
plt.show()

```

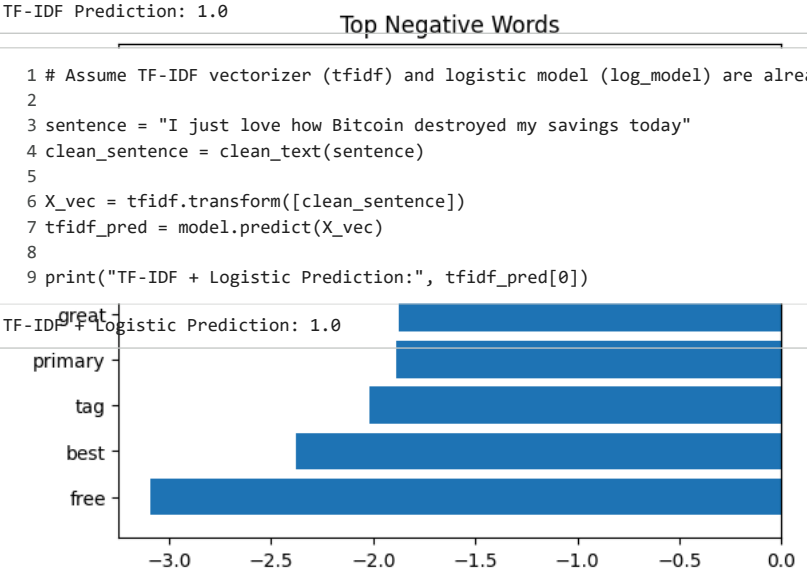



```
import numpy as np

feature_names = tfidf.get_feature_names_out()
coef = model.coef_

# POSITIVE class (highest weights)
top_positive = np.argsort(coef[2])[-10:]
plt.barh(feature_names[top_positive], coef[2][top_positive])
plt.title("Top Positive Words")
plt.show()

# NEGATIVE class (lowest weights)
top_negative = np.argsort(coef[0])[:10]
plt.barh(feature_names[top_negative], coef[0][top_negative])
plt.title("Top Negative Words")
plt.show()

```




```
# TF-IDF + Logistic Regression prediction

user_input = ["I just lost all my money in Bitcoin, great job crypto"]
user_clean = [clean_text(user_input[0])]

user_tfidf = tfidf.transform(user_clean)
tfidf_pred = model.predict(user_tfidf)

print("TF-IDF Prediction:", tfidf_pred[0])

```

```
TF-IDF Prediction: 1.0
```




```
# Assume TF-IDF vectorizer (tfidf) and logistic model (log_model) are already trained

sentence = "I just love how Bitcoin destroyed my savings today"
clean_sentence = clean_text(sentence)

X_vec = tfidf.transform([clean_sentence])
tfidf_pred = model.predict(X_vec)

print("TF-IDF + Logistic Prediction:", tfidf_pred[0])
```

```
TF-IDF + Logistic Prediction: 1.0
```