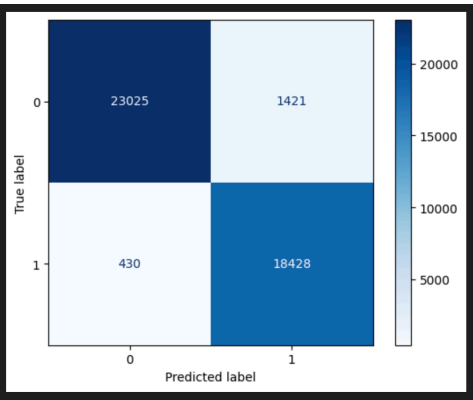



```
import joblib
final_dff.to_csv("/kaggle/working/final_dff.csv", index=False)
print("Original dataset saved as final_dff.csv")


# Save Models
```

Python

```
Vectorization done. Train shape: (157086, 5000)
```

XG

```
xgb_model = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss'
)

xgb_model.fit(X_train, y_train)
```

Python

```
# Predictions
y_pred_xgb = xgb_model.predict(X_train)

# Evaluation
print("XGB Metrics:")
print("Accuracy:", accuracy_score(y_train, y_pred_xgb))
print("Precision:", precision_score(y_train, y_pred_xgb))
print("Recall:", recall_score(y_train, y_pred_xgb))
print("F1 Score:", f1_score(y_train, y_pred_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_train, y_pred_xgb))
```

Python

estricted Mode ⊗ 0 ⚠ 17 Spaces: 4 Cell 7 of 44 Prettier

```
import numpy as np

# Separate classes
df_ham = df_train_all[df_train_all['label'] == 0]
df_spam = df_train_all[df_train_all['label'] == 1]

# Find how many to duplicate
if len(df_ham) > len(df_spam):
    diff = len(df_ham) - len(df_spam)
    df_spam_oversampled = pd.concat([df_spam, df_spam.sample(diff, replace=True, random_state=42)])
    df_balanced = pd.concat([df_ham, df_spam_oversampled])
else:
    diff = len(df_spam) - len(df_ham)
    df_ham_oversampled = pd.concat([df_ham, df_ham.sample(diff, replace=True, random_state=42)])
    df_balanced = pd.concat([df_ham_oversampled, df_spam])

# Shuffle
df_balanced = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

# Check new distribution
new_counts = df_balanced['label'].value_counts()
new_percent = (new_counts / new_counts.sum() * 100).round(2)
print("Balanced label counts:\n", new_counts)
print("Balanced percentage distribution:\n", new_percent)
```

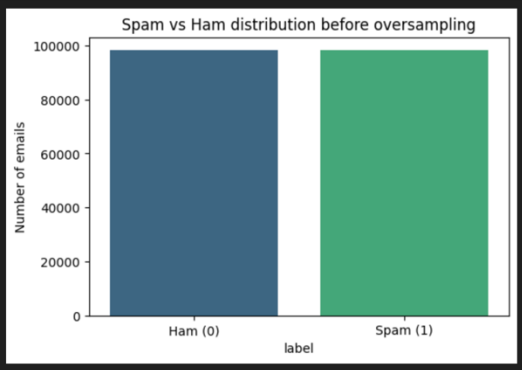
Python

```
Balanced label counts:
 label
1    98179
0    98179
Name: count, dtype: int64
Balanced percentage distribution:
```

cted Mode ⊗ 0 ⚠ 17 Spaces: 4 Cell 7 of 44 Prettier

```python
# Original label distribution
label_counts = df_balanced['label'].value_counts()
plt.figure(figsize=(6,4))
sns.barplot(x=label_counts.index, y=label_counts.values, palette="viridis")
plt.xticks([0,1], ['Ham (0)', 'Spam (1)'])
plt.ylabel("Number of emails")
plt.title("Spam vs Ham distribution before oversampling")
plt.show()
```

Pytho




```python
y_pred_xgb_test = xgb_model.predict(X_test_final)

print("XGBoost on new test data:")
print("Accuracy:", accuracy_score(y_test_final, y_pred_xgb_test))
print("Precision:", precision_score(y_test_final, y_pred_xgb_test))
print("Recall:", recall_score(y_test_final, y_pred_xgb_test))
print("F1 Score:", f1_score(y_test_final, y_pred_xgb_test))
print("Confusion Matrix:\n", confusion_matrix(y_test_final, y_pred_xgb_test))
```

[31] Python

```
XGBoost on new test data:
Accuracy: 0.9572556807685202
Precision: 0.9284094916620484
Recall: 0.9771980061512355
F1 Score: 0.9521791923941406
Confusion Matrix:
 [[23025  1421]
 [  430 18428]]
```

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Compute confusion matrix
cm = confusion_matrix(y_test_final, y_pred_xgb_test)

# Display it
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')  # Optional: color map
```

[32] Python

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x78245927e3d0>
```

```python
print("Accuracy:", accuracy_score(y_train, y_pred_xgb))
print("Precision:", precision_score(y_train, y_pred_xgb))
print("Recall:", recall_score(y_train, y_pred_xgb))
print("F1 Score:", f1_score(y_train, y_pred_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_train, y_pred_xgb))
```

Python

```
XGB Metrics:
Accuracy: 0.9683230841704544
Precision: 0.9516471642744005
Recall: 0.9867843092318858
F1 Score: 0.9688972785118699
Confusion Matrix:
 [[74605  3938]
 [ 1038 77505]]
```

```python
# Predictions
y_pred_xgb = xgb_model.predict(X_test)

# Evaluation
print("XGBoost Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("Precision:", precision_score(y_test, y_pred_xgb))
print("Recall:", recall_score(y_test, y_pred_xgb))
print("F1 Score:", f1_score(y_test, y_pred_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))
```

Python

```
XGBoost Metrics:
Accuracy: 0.9617539213689142
Precision: 0.9445043631728601
Recall: 0.9811570584640457
```



```python
joblib.dump(xgb_model, "/kaggle/working/xgb_model.pkl")
print("XGBoost model saved as xgb_model.pkl")

#Save Vectorizer
joblib.dump(vectorizer, "/kaggle/working/tfidf_vectorizer.pkl")
print("TF-IDF vectorizer saved as tfidf_vectorizer.pkl")
```

Python

```
Original dataset saved as final_dff.csv
XGBoost model saved as xgb_model.pkl
TF-IDF vectorizer saved as tfidf_vectorizer.pkl
```

```python
# Approximate sample size in rows
# (since exact MB depends on number of columns & text length)
sample_fraction = 24 * 1024 * 1024 / final_dff.memory_usage(deep=True).sum()

# Take random sample
df_sample = final_dff.sample(frac=sample_fraction, random_state=42)

# Save to new CSV (~24 MB)
df_sample.to_csv("sample_24mb.csv", index=False)
```

Python

```python
from IPython.display import FileLink

# Make a clickable download link in Kaggle notebook
FileLink("sample_24mb.csv")
```

Python