```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from catboost import CatBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import Normalizer, MinMaxScaler
from sentence_transformers import SentenceTransformer

# Load training and test data
train_file = 'Train.csv'
test_file = 'Test.csv'
output_file = 'submissions.csv'

# Training data does not have a header
train_data = pd.read_csv(train_file, header=None, names=['text',
'subreddit'])
test_data = pd.read_csv(test_file)

# Preprocessing metadata (TF-IDF and N-grams)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),
max_features=5000)
tfidf_features = tfidf_vectorizer.fit_transform(train_data['text'])

# Perform dimensionality reduction on TF-IDF using TruncatedSVD
svd = TruncatedSVD(n_components=300, random_state=42)
reduced_tfidf = svd.fit_transform(tfidf_features)

# Ensure non-negative features for MultinomialNB
minmax_scaler = MinMaxScaler()
non_negative_tfidf = minmax_scaler.fit_transform(reduced_tfidf)

# Normalize TF-IDF features for Logistic Regression and CatBoost
tfidf_normalizer = Normalizer(norm='l2')
normalized_train_tfidf = tfidf_normalizer.fit_transform(reduced_tfidf)

# Load Sentence Transformer model
sentence_model = SentenceTransformer('paraphrase-multilingual-MiniLM-
L12-v2')
sentence_embeddings =
sentence_model.encode(train_data['text'].tolist())

# Normalize Sentence Embeddings
sentence_normalizer = Normalizer(norm='l2')
normalized_train_sentences =
sentence_normalizer.fit_transform(sentence_embeddings)

# Combine features (Normalized TF-IDF + Normalized Sentence
```

```
Embeddings)
X_combined = np.hstack([
    normalized_train_tfidf,              # Normalized TF-IDF features
(300 dims)
    normalized_train_sentences           # Normalized Sentence
Embeddings (84 dims)
])

# Map labels
label_map = {label: idx for idx, label in
enumerate(train_data['subreddit'].unique())}
y = train_data['subreddit'].map(label_map)

# Hyperparameter grids
param_grid_nb = {
    'alpha': [0.01, 0.1, 1.0]
}

param_grid_lr = {
    'C': [0.1, 1, 10],
    'solver': ['lbfgs']
}

param_grid_cb = {
    'iterations': [100, 200],
    'learning_rate': [0.05],
    'depth': [4, 6]
}

# Multinomial Naive Bayes
nb_model = MultinomialNB()
grid_search_nb = GridSearchCV(
    nb_model, param_grid=param_grid_nb, cv=3, scoring='accuracy',
verbose=1, n_jobs=-1
)
grid_search_nb.fit(non_negative_tfidf, y)

# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
grid_search_lr = GridSearchCV(
    lr_model, param_grid=param_grid_lr, cv=3, scoring='accuracy',
verbose=1, n_jobs=-1
)
grid_search_lr.fit(X_combined, y)

# CatBoost
cb_model = CatBoostClassifier(verbose=0)
grid_search_cb = GridSearchCV(
    cb_model, param_grid=param_grid_cb, cv=3, scoring='accuracy',
verbose=1, n_jobs=-1
```

```
)
grid_search_cb.fit(X_combined, y)

# Save best estimators
nb_best_model = grid_search_nb.best_estimator_
lr_best_model = grid_search_lr.best_estimator_
cb_best_model = grid_search_cb.best_estimator_

# Metrics
metrics = {
    'Classifier': ['Multinomial Naive Bayes', 'Logistic Regression',
'CatBoost'],
    'Training Accuracy': [
        grid_search_nb.best_score_,
        grid_search_lr.best_score_,
        grid_search_cb.best_score_
    ]
}
metrics_df = pd.DataFrame(metrics)
print(metrics_df)

# Process test set: TF-IDF
test_tfidf_features = tfidf_vectorizer.transform(test_data['body'])
test_reduced_tfidf = svd.transform(test_tfidf_features)  # Reduce
dimensions to 300
test_normalized_tfidf = tfidf_normalizer.transform(test_reduced_tfidf)
# Normalize TF-IDF

# Process test set: Sentence Embeddings
test_sentence_embeddings =
sentence_model.encode(test_data['body'].tolist())
test_normalized_sentence_embeddings =
sentence_normalizer.transform(test_sentence_embeddings)  # Normalize
Sentence Embeddings

# Combine test features (TF-IDF + Sentence Embeddings)
test_combined = np.hstack([
    test_normalized_tfidf,              # Normalized TF-IDF features
(300 dims)
    test_normalized_sentence_embeddings  # Normalized Sentence
Embeddings (84 dims)
])

# Predict on test set using best CatBoost model
test_predictions = cb_best_model.predict(test_combined)

# Map predictions back to labels
reverse_label_map = {idx: label for label, idx in label_map.items()}

# Flatten predictions and map back to labels
```

```
test_predictions = test_predictions.flatten() if
len(test_predictions.shape) > 1 else test_predictions
test_data['subreddit'] = [reverse_label_map[int(pred)] for pred in
test_predictions]

# Create submission file
submission = test_data[['id', 'subreddit']]
submission.to_csv(output_file, index=False)
print(f"Submission file saved as: {output_file}")
```

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
Fitting 3 folds for each of 3 candidates, totalling 9 fits
Fitting 3 folds for each of 4 candidates, totalling 12 fits
                Classifier  Training Accuracy
0  Multinomial Naive Bayes           0.589274
1      Logistic Regression           0.731398
2                 CatBoost           0.684257
Submission file saved as: submissions.csv
```