Best Pipeline with Larger Sample

All 3 pipelines have almost the same ROC-AUC Accuracy scoring. The AUC score of both training and testing data for all the 3 pipelines is closeby which shows it is not much overfitting, The confusion matrix gives a much better picture about the accuracy of the models. The 3rd Pipeline is much better at predicting SPAM emails as compared to others.

Pipeline 3 is better at predicting than other pipelines.

```
# Connect Google colab to drive
from pathlib import Path
if 'google.colab' in str(get_ipython()):
  from google.colab import drive
  drive.mount('/content/drive')
  base_folder = Path('/content/drive/MyDrive/NLP')
data_folder = base_folder/'Datasets'
```

```
Mounted at /content/drive
```

```
#Load the pandas dataframe
data = data_folder/'spam.csv'
import pandas as pd
spam_data = pd.read_csv(data,encoding='latin-1')
spam_data = spam_data.drop(columns = ['Unnamed: 2','Unnamed: 3','Unnamed: 4'])
#Rename the columns
spam_data.columns = ['Label' , 'Message']
```

```
import re
from bs4 import BeautifulSoup
def cleaned_text(text):
  #Remove HTML tags
  clean_text = BeautifulSoup(text , "html.parser").get_text()
  #Remove new line characters and replace them with space
  clean_text = re.sub(r'[\n\r]',' ', clean_text)
  #Remove URLs:
  clean_text = re.sub(r'http\S+' , '' , clean_text)
  #Remove Emails:
  clean_text = re.sub(r'\S+@\S+' , '' , clean_text)
  #Remove punctuation:
  clean_text = re.sub(r'[^\w\s.]' , '' , clean_text)
  return clean_text
```

```
#Extracting larger sample from spam data
large_sample = spam_data.sample(frac = 0.60 , random_state = 11)
large_sample['cleaned_text'] = large_sample['Message'].apply(cleaned_text)
```

```
<ipython-input-4-b2a3c14291fa>:5: MarkupResemblesLocatorWarning: The input looks more like a filename than markup. You may want to
  clean_text = BeautifulSoup(text , "html.parser").get_text()
```

```
large_sample.head()
```

	Label	Message	cleaned_text
4460	ham	Thanks again for your reply today. When is ur ...	Thanks again for your reply today. When is ur ...
1049	spam	18 days to Euro2004 kickoff! U will be kept in...	18 days to Euro2004 kickoff U will be kept inf...
1810	ham	Now, whats your house # again ? And do you hav...	Now whats your house again And do you have a...
		YOUR CHANCE TO BE ON A	YOUR CHANCE TO BE ON A REALITY

```
#Import spacy
import spacy
nlp = spacy.load('en_core_web_sm')
```

```
#Feature Extraction-
disabled = nlp.select_pipes(disable=['lemmatizer','ner'])
entities_count = []
symbol_count = []
punct_count = []
num_count = []
uppercase_count = []
```

```
special_char_count = []
url_email_count = []
for doc in nlp.pipe(large_sample.Message.values , batch_size = 1000):
    entities = [token.text for token in doc if (token.pos_ in ['PROPN'])]
    entities_count.append(len(entities))
    symbols = [token.text for token in doc if (token.pos_ in ['SYM'])]
    symbol_count.append(len(symbols))
    punct = [token.text for token in doc if (token.pos_ in ['PUNCT'])]
    punct_count.append(len(punct))
    number = [token.text for token in doc if (token.pos_ in ['NUM'])]
    num_count.append(len(number))
    uppercase = [token.text for token in doc if (token.is_upper)]
    uppercase_count.append(len(uppercase))
    special_char = [token.text for token in doc if (c.isalpha() and not c.isascii() for c in token.text)]
    special_char_count.append(len(special_char))
    url_email = [token.text for token in doc if (token.like_url or token.like_email)]
    url_email_count.append(len(url_email))
disabled.restore()
```

```
large_sample2 = large_sample.copy()
large_sample2['Entity_count'] = entities_count
large_sample2['Symbol_count'] = symbol_count
large_sample2['Punct_count'] = punct_count
large_sample2['Num_count'] = num_count
large_sample2['Uppercase_count'] = uppercase_count
large_sample2['Special_char_count'] = special_char_count
large_sample2['Url_email_count'] = url_email_count
large_sample2.head()
```

	Label	Message	cleaned_text	Entity_count	Symbol_count	Punct_count	Num_count	Uppercase_count	Special_char_count	Url_em
4460	ham	Thanks again for your reply today. When is ur ...	Thanks again for your reply today. When is ur ...	3	0	6	0	0	55	
1049	spam	18 days to Euro2004 kickoff! U will be kept in...	18 days to Euro2004 kickoff U will be kept inf...	3	0	3	2	4	28	
1810	ham	Now, whats your house # again ? And do	Now whats your house again And do you have a...	0	0	3	0	0	17	

```
#Train-Test Split-
df_final = large_sample2.drop(columns = ['Message'])
X_train_final = df_final.drop(columns = ['Label']).iloc[:1672,]
X_test_final = df_final.drop(columns = ['Label']).iloc[1672:,]
label_map = {"ham": 0, "spam": 1}
y_train_final = df_final['Label'].iloc[:1672,].map(label_map)
y_test_final = df_final['Label'].iloc[1672:,].map(label_map)
```

```
#Pipeline Building
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = Pipeline([('tfidf', TfidfVectorizer(max_features=5)), ])
combined_features = ColumnTransformer(
    transformers=[
        ('tfidf', vectorizer, 'cleaned_text'),
    ],remainder='passthrough'
)
```

```
#Final Classifier
import xgboost
from xgboost import XGBClassifier
classifier_3 = Pipeline([('combined_features',  combined_features),
                         ('classifier', XGBClassifier()),
                         ])
```

```
#Parameters for GridSearchCV
param_grid_classifier_3 = {'combined_features__tfidf__tfidf__max_features': [500, 1000, 2000],
```

```
                         'classifier__learning_rate': [0.01, 0.1, 0.2]
                         }


#GridSearchCV
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
grid_classifier_3 = GridSearchCV(estimator=classifier_3,
                                 param_grid=param_grid_classifier_3,
                                 cv=3)


#Using the Final Pipeline with GridSearchCV
grid_classifier_3.fit(X_train_final, y_train_final)
```

```
GridSearchCV
estimator: Pipeline
combined_features: ColumnTransformer
tfidf                remainder
TfidfVectorizer      passthrough
XGBClassifier
```

```
#Best Parameters -
grid_classifier_3.best_params_
```

```
{'classifier__learning_rate': 0.2,
 'combined_features__tfidf__tfidf__max_features': 500}
```

```
#Accuracy Score
from sklearn.model_selection import cross_val_score
training_scores_final = cross_val_score(grid_classifier_3.best_estimator_, X_train_final, y_train_final, scoring='roc_auc', cv=3, n_jobs=
testing_scores_final = cross_val_score(grid_classifier_3.best_estimator_, X_test_final , y_test_final , cv = 3 , scoring = 'roc_auc', n_
print('Mean Training ROC AUC for Final Pipeline: %.5f' % training_scores_final.mean())
print('Mean Testing ROC AUC for Final Pipeline 3: %.5f' % testing_scores_final.mean())
```

```
Mean Training ROC AUC for Final Pipeline: 0.98243
Mean Testing ROC AUC for Final Pipeline 3: 0.96301
```

```
model_final = grid_classifier_3.best_estimator_
y_pred_final = model_final.predict(X_test_final)


#Classification Report-
from sklearn.metrics import classification_report,confusion_matrix
print('\nTest set classification report for Final Pipeline:\n\n',
      classification_report(y_test_final, y_pred_final))
```

```
Test set classification report for Final Pipeline:

               precision    recall  f1-score   support

           0       0.97      1.00      0.98      1440
           1       0.96      0.80      0.87       231

    accuracy                           0.97      1671
   macro avg       0.97      0.90      0.93      1671
weighted avg       0.97      0.97      0.97      1671
```

```
#Confusion Matrix-
cm_final = confusion_matrix(y_test_final, y_pred_final)

# Print the confusion matrix
print("Confusion Matrix:")
print(cm_final)
```

```
Confusion Matrix:
[[1433    7]
 [  46  185]]
```