# Import CountVectorizer

from sklearn.feature\_extraction.text import CountVectorize

# Create the token pattern: TOKENS\_ALPHANUMERIC

TOKENS\_ALPHANUMERIC = '[A-Za-z0-9]+(?=\\s+)'

# Fill missing values in df.Position\_Extra

df.Position\_Extra.fillna('',inplace = True)

# Instantiate the CountVectorizer: vec\_alphanumeric

vec\_alphanumeric = CountVectorize(token\_pattern = TOKENS\_ALPHANUMERIC)

# Fit to the data

vec\_alphanumeric.fit(df.Position\_Extra)

# Print the number of tokens and first 15 tokens

msg = "There are {} tokens in Position\_Extra if we split on non-alpha numeric"

print(msg.format(len(vec\_alphanumeric.get\_feature\_names())))

print(vec\_alphanumeric.get\_feature\_names()[:15])

# Import the CountVectorizer

from sklearn.feature\_extraction.text import CountVectorizer

# Split out only the text data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(sample\_df['text'],

pd.get\_dummies(sample\_df['label']),

random\_state=456)

# Instantiate Pipeline object: pl

pl = Pipeline([

('vec', CountVectorizer()),

('clf', OneVsRestClassifier(LogisticRegression()))

])

# Fit to the training data

pl.fit(X\_train,y\_train)

# Compute and print accuracy

accuracy = pl.score(X\_test,y\_test)

print("\nAccuracy on sample data - just text data: ", accuracy)

# Import FunctionTransformer

from sklearn.preprocessing import FunctionTransformer

# Obtain the text data: get\_text\_data

get\_text\_data = FunctionTransformer(lambda x: x['text'], validate=False)

# Obtain the numeric data: get\_numeric\_data

get\_numeric\_data = FunctionTransformer(lambda x: x[['numeric', 'with\_missing']], validate=False)

# Fit and transform the text data: just\_text\_data

just\_text\_data = get\_text\_data.fit\_transform(sample\_df)

# Fit and transform the numeric data: just\_numeric\_data

just\_numeric\_data = get\_numeric\_data.fit\_transform(sample\_df)

# Print head to check results

print('Text Data')

print(just\_text\_data.head())

print('\nNumeric Data')

print(just\_numeric\_data.head())

# Import FeatureUnion

from sklearn.pipeline import FeatureUnion

# Split using ALL data in sample\_df

X\_train, X\_test, y\_train, y\_test = train\_test\_split(sample\_df[['numeric', 'with\_missing', 'text']],

pd.get\_dummies(sample\_df['label']),

random\_state=22)

# Create a FeatureUnion with nested pipeline: process\_and\_join\_features

process\_and\_join\_features = FeatureUnion(

transformer\_list = [

('numeric\_features', Pipeline([

('selector', get\_numeric\_data),

('imputer', Imputer())

])),

('text\_features', Pipeline([

('selector', get\_text\_data),

('vectorizer', CountVectorizer())

]))

]

)

# Instantiate nested pipeline: pl

pl = Pipeline([

('union', process\_and\_join\_features),

('clf', OneVsRestClassifier(LogisticRegression()))

])

# Fit pl to the training data

pl.fit(X\_train, y\_train)

# Compute and print accuracy

accuracy = pl.score(X\_test, y\_test)

print("\nAccuracy on sample data - all data: ", accuracy)

# Instantiate pipeline: pl

pl = Pipeline([

('union', FeatureUnion(

transformer\_list = [

('numeric\_features', Pipeline([

('selector', get\_numeric\_data),

('imputer', Imputer())

])),

('text\_features', Pipeline([

('selector', get\_text\_data),

('vectorizer', CountVectorizer(token\_pattern=TOKENS\_ALPHANUMERIC,

ngram\_range=(1, 2))),

('dim\_red', SelectKBest(chi2, chi\_k))

]))

]

)),

('int', SparseInteractions(degree=2)),

('scale', MaxAbsScaler()),

('clf', OneVsRestClassifier(LogisticRegression()))

])

# Import HashingVectorizer

from sklearn.feature\_extraction.text import HashingVectorizer

# Get text data: text\_data

text\_data = combine\_text\_columns(X\_train)

# Create the token pattern: TOKENS\_ALPHANUMERIC

TOKENS\_ALPHANUMERIC = '[A-Za-z0-9]+(?=\\s+)'

# Instantiate the HashingVectorizer: hashing\_vec

hashing\_vec = HashingVectorizer(token\_pattern = TOKENS\_ALPHANUMERIC )

# Fit and transform the Hashing Vectorizer

hashed\_text = hashing\_vec.fit\_transform(text\_data)

# Create DataFrame and print the head

hashed\_df = pd.DataFrame(hashed\_text.data)

print(hashed\_df.head())

# Import the hashing vectorizer

from sklearn.feature\_extraction.text import HashingVectorizer

# Instantiate the winning model pipeline: pl

pl = Pipeline([

('union', FeatureUnion(

transformer\_list = [

('numeric\_features', Pipeline([

('selector', get\_numeric\_data),

('imputer', Imputer())

])),

('text\_features', Pipeline([

('selector', get\_text\_data),

('vectorizer', HashingVectorizer(token\_pattern=TOKENS\_ALPHANUMERIC,

non\_negative=True, norm=None, binary=False,

ngram\_range=(1, 2)))),

('dim\_red', SelectKBest(chi2, chi\_k))

]))

]

)),

('int', SparseInteractions(degree=2)),

('scale', MaxAbsScaler()),

('clf', OneVsRestClassifier(LogisticRegression()))

])