Machine Learning with the Experts: School Budgets

4). Learning from the experts:

a). Deciding what's a word

```
# Import the CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer

# Create the text vector
text_vector = combine_text_columns(X_train)

# Create the token pattern: TOKENS_ALPHANUMERIC
TOKENS_ALPHANUMERIC = '[A-Za-z0-9]+(?=\\s+)'

# Instantiate the CountVectorizer: text_features
text_features = CountVectorizer(token_pattern=TOKENS_ALPHANUMERIC)

# Fit text_features to the text vector
text_features.fit(text_vector)

# Print the first 10 tokens
print(text_features.get_feature_names()[:10])
```

```
<script.py> output:
    ['00a', '12', '1st', '2nd', '3rd', '5th', '70', '70h', '8', 'a']
```

b). N-gram range in sci-kit learn

```
# Import pipeline
from sklearn.pipeline import Pipeline

# Import classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

# Import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer

# Import other preprocessing modules
from sklearn.preprocessing import Imputer
from sklearn.feature_selection import chi2, SelectKBest

# Select 300 best features
chi_k = 300

# Import functional utilities
from sklearn.preprocessing import FunctionTransformer, MaxAbsScaler
from sklearn.pipeline import FeatureUnion

# Perform preprocessing
get_text_data = FunctionTransformer(combine_text_columns, validate=False)
get_numeric_data = FunctionTransformer(lambda x: x[NUMERIC_COLUMNS], validate=False)

# Create the token pattern: TOKENS_ALPHANUMERIC
TOKENS_ALPHANUMERIC = '[A-Za-z0-9]+(?=\\s+)'

# 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))
            ]))
        ]
    )),
    ('scale', MaxAbsScaler()),
    ('clf', OneVsRestClassifier(LogisticRegression()))
])
```

c). <u>Implement Interaction modeling in scikit learn</u>

```
# 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()))
    ])
```

d). Implementing the hashing trick in scikit-learn

```
# 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())
```

<script.py> output:

```
          0
0 -0.160128
1  0.160128
2 -0.480384
3 -0.320256
4  0.160128
```

e). **Build the winning model**

```
# 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()))
  ])
```