Machine Learning with the Experts: School Budgets

3). Improving your model:

a). Instantiate pipeline

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

# Import other necessary modules
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

# Split and select numeric data only, no nans
X_train, X_test, y_train, y_test = train_test_split(sample_df[['numeric']],
pd.get_dummies(sample_df['label']), random_state=22)

# Instantiate Pipeline object: pl
pl = Pipeline([
        ('clf', OneVsRestClassifier(LogisticRegression()))
    ])

# Fit the pipeline 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 - numeric, no nans: ", accuracy)
```

<script.py> output:

```
Accuracy on sample data - numeric, no nans:  0.62
```

b). Preprocessing Numeric Features

```
# Import the Imputer object
from sklearn.preprocessing import Imputer


# Create training and test sets using only numeric data
X_train, X_test, y_train, y_test = train_test_split(sample_df[['numeric', 'with_missing']],
                                    pd.get_dummies(sample_df['label']),
                                    random_state=456)


# Insantiate Pipeline object: pl
pl = Pipeline([
      ('imp', Imputer()),
      ('clf', OneVsRestClassifier(LogisticRegression()))
   ])


# Fit the pipeline 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 numeric, incl nans: ", accuracy)
```

<script.py> output:

```
Accuracy on sample data - all numeric, incl nans:  0.636
```

c). Preprocessing text features

```
# 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)
```

<script.py> output:

```
Accuracy on sample data - just text data:  0.808
```

d). <u>Multiple Type of Processing Function Transformer</u>

```
# 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())
```

<script.py> output:

Text Data

```
0
1      foo
2  foo bar
3
4  foo bar
Name: text, dtype: object
```

Numeric Data

```
    numeric  with_missing
0 -10.856306      4.433240
1   9.973454      4.310229
2   2.829785      2.469828
3 -15.062947      2.852981
4  -5.786003      1.826475
```

e). <u>Multiple type of Processing: Feature Union</u>

```
# 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)
```

f). <u>Using Function Transformer on the main dataset</u>

```
# Import FunctionTransformer
from sklearn.preprocessing import FunctionTransformer

# Get the dummy encoding of the labels
dummy_labels = pd.get_dummies(df[LABELS])

# Get the columns that are features in the original df
NON_LABELS = [c for c in df.columns if c not in LABELS]

# Split into training and test sets
X_train, X_test, y_train, y_test = multilabel_train_test_split(df[NON_LABELS], dummy_labels, 0.2,
seed=123)

# Preprocess the text data: get_text_data
get_text_data = FunctionTransformer(combine_text_columns, validate=False)

# Preprocess the numeric data: get_numeric_data
get_numeric_data = FunctionTransformer(lambda x: x[NUMERIC_COLUMNS], validate=False)
```

g). Add a model to the pipeline

```
# Complete the 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())
                ]))
            ]
        )),
        ('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 budget dataset: ", accuracy)
```

<script.py> output:

```
    Accuracy on budget dataset:  0.203846153846
```

h). Trying Random Forest Classifier

```
# Import random forest classifer
from sklearn.ensemble import RandomForestClassifier

# Edit model step in pipeline
pl = Pipeline([
        ('union', FeatureUnion(
            transformer_list = [
                ('numeric_features', Pipeline([
                    ('selector', get_numeric_data),
                    ('imputer', Imputer())
                ])),
                ('text_features', Pipeline([
                    ('selector', get_text_data),
                    ('vectorizer', CountVectorizer())
                ]))
            ]
        )),
        ('clf', RandomForestClassifier())
    ])

# 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 budget dataset: ", accuracy)
```

<script.py> output:

```
Accuracy on budget dataset:  0.296153846154
```

i). <u>Improving Model or Parameter to improve accuracy:</u>

```
# Import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

# Add model step to 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())
              ]))
           ]
        )),
        ('clf', RandomForestClassifier(n_estimators=15))
    ])
# 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 budget dataset: ", accuracy)
```

<script.py> output:

```
    Accuracy on budget dataset:  0.346153846154
```