Python Models

May 9, 2022

[117]:
```
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score
```

[115]:
```
catColumns = ['AntiSJW','AntiTheist', 'Black', 'Conspiracy', 'LGBT',
       'LateNightTalkShow', 'Libertarian', 'MRA', 'PartisanLeft',␣
↪'PartisanRight', 'Politician', 'QAnon',
       'ReligiousConservative', 'SocialJustice', 'Socialist', 'StateFunded',
       'WhiteIdentitarian', "C", "L", "R"]
```

[6]:
```
def scale_feature(df, column_name):
    minviews = min(df[column_name])
    maxviews = max(df[column_name])
    def feature_scaling(x):
        return (x - minviews)/(maxviews - minviews)
    df[f'scaled_{column_name}'] = df[column_name].apply(feature_scaling)
```

[112]:
```
def getData(splitType, splitSize, relevantColumns):
    dataPath = f"{rootPath}/{splitType}/{splitSize}"
    train = pd.read_csv(f"{dataPath}/X_train_bert.csv", index_col=0)
    Y_train = pd.read_csv(f"{dataPath}/y_train.csv", index_col=0)
    test = pd.read_csv(f"{dataPath}/X_test_bert.csv", index_col=0)
    Y_test = pd.read_csv(f"{dataPath}/y_test.csv", index_col=0)
    train[list(dict(train.groupby("lr").size()))] = pd.get_dummies(train["lr"])
    test[list(dict(test.groupby("lr").size()))] = pd.get_dummies(test["lr"])
    train[list(dict(train.groupby("channelId").size()))] = pd.
↪get_dummies(train["channelId"])
    test[list(dict(test.groupby("channelId").size()))] = pd.
↪get_dummies(test["channelId"])
    X = train[relevantColumns].to_numpy()
    y = np.array(list(Y_train["interesting_removal_2"].astype(int)))
    X_test = test[relevantColumns].to_numpy()
    y_test = np.array(list(Y_test["interesting_removal_2"].astype(int)))
#     X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
    return X, X_test, y, y_test
```

[91]:

```
def generateLabels():
    EMBED_OUTPUT = 768
    labels = []
    for i in range(EMBED_OUTPUT):
        labels.append(f"embed_{i+1}")
    return labels
```

[131]:

```
def evaluateModel(model, x, y):
    metric = {

    }
    y_pred = model.predict(x)
    metric["accuracy"] = np.sum(y_pred == y)/y.shape[0]
    metric["precision"] = precision_score(y_pred, y)
    metric["recall"] = recall_score(y_pred, y)
    return metric
```

[145]:

```
splits = ["rep", "even"]
sizes = ["10", "50", "100"]
```

[152]:

```
# relevantColumns = catColumns
relevantColumns = catColumns + generateLabels()
```

[153]:

```
for split in splits:
    for size in sizes:
        X_train, X_test, y_train, y_test = getData(split, size, relevantColumns)
#         clf = SVC(kernel='poly')
        clf = MLPClassifier(solver='lbfgs', alpha=1e-4, activation="relu",␣
↪hidden_layer_sizes=(160), random_state=1)
        clf.fit(X_train, y_train)
        train_accuracy = evaluateModel(clf, X_train, y_train)
        test_accuracy = evaluateModel(clf, X_test, y_test)
        print(f"Split: {split}")
        print(f"Size: {size}")
        print(train_accuracy)
        print(test_accuracy)
```

```
Split: rep
Size: 10
{'accuracy': 0.9996473906911142, 'precision': 0.9919354838709677, 'recall': 1.0}
{'accuracy': 0.956338028169014, 'precision': 0.3548387096774194, 'recall': 0.5}
Split: rep
```

```
Size: 50
{'accuracy': 0.9997179921037789, 'precision': 0.9965635738831615, 'recall':
0.9965635738831615}
{'accuracy': 0.9625035241048774, 'precision': 0.4330708661417323, 'recall':
0.47413793103448276}
Split: rep
Size: 100
{'accuracy': 0.9994359842075579, 'precision': 0.9939498703543648, 'recall':
0.9922346850733391}
{'accuracy': 0.9641900465247427, 'precision': 0.4925373134328358, 'recall':
0.528}
Split: even
Size: 10
{'accuracy': 1.0, 'precision': 1.0, 'recall': 1.0}
{'accuracy': 0.6896551724137931, 'precision': 0.7586206896551724, 'recall':
0.6666666666666666}
Split: even
Size: 50
{'accuracy': 1.0, 'precision': 1.0, 'recall': 1.0}
{'accuracy': 0.8140350877192982, 'precision': 0.823943661971831, 'recall':
0.8068965517241379}
Split: even
Size: 100
{'accuracy': 0.9991228070175439, 'precision': 1.0, 'recall': 0.9982486865148862}
{'accuracy': 0.8192982456140351, 'precision': 0.8245614035087719, 'recall':
0.8159722222222222}
```