initial_ff_testing

February 28, 2024

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from numpy import random
```

[ ]:
```
random.seed(42)

data = pd.read_csv('data/Features_For_Traditional_ML_Techniques.csv')
#remove all meta-data features as we are comparing structure (except hashtags)
#compare to original data performance on baseline models
meta_features = ["followers_count",␣
 ↪"friends_count","favourites_count","statuses_count","listed_count","following","embeddings"
#filter data to only include non-meta features
data = data.drop(meta_features, axis=1)

data = data.drop('majority_target', axis=1)
```

[ ]:
```
# group by statements
# will change when get subcategories
grouped = data.groupby('statement')
train_data = pd.DataFrame()
test_data = pd.DataFrame()
random.seed(42)

i=0

for group_name, group_df in grouped:
    if random.random() <= 0.8:
        train_data = pd.concat([train_data, group_df])
    else:
        test_data = pd.concat([test_data, group_df])
    i += 1
```

[ ]:
```
len(train_data), len(test_data)
train_statement_unique = set(train_data["statement"].unique())
test_statement_unique = set(test_data["statement"].unique())

common_elements = train_statement_unique.intersection(test_statement_unique)
if len(common_elements) != 0:
    print("Error: common elements between train and test data")
    print(len(common_elements))
else:
    print("No common elements between train and test data")
```

```
No common elements between train and test data
```

[ ]:
```
import torch as torch
import torch.nn as nn
```

[ ]:
```
params = {}
params['input_size'] = train_data.iloc[0].shape[0] - 1 #46 features after removing target
params['hidden_size'] = 100 # arbitrary
params['num_classes'] = 2 # binary classification, real or fake
params['num_epochs'] = 10
params['batch_size'] = 64
params['learning_rate'] = 0.001
```

[ ]:
```
print(train_data.iloc[0])
```

```
Unnamed: 0                                                        68447
statement                 "(M)ore Georgians have jobs than at any other …
BinaryNumTarget                                                     1.0
tweet                     @GovernorDeal More Georgians have jobs than at…
hashtags                                                            0.0
unique_count                                                          2
total_count                                                           2
ORG_percentage                                                      0.5
NORP_percentage                                                     0.0
GPE_percentage                                                      0.0
PERSON_percentage                                                   0.0
MONEY_percentage                                                    0.0
DATE_percentage                                                     0.0
CARDINAL_percentage                                                 0.5
PERCENT_percentage                                                  0.0
ORDINAL_percentage                                                  0.0
FAC_percentage                                                      0.0
LAW_percentage                                                      0.0
PRODUCT_percentage                                                  0.0
EVENT_percentage                                                    0.0
TIME_percentage                                                     0.0
```

```
LOC_percentage                                                  0.0
WORK_OF_ART_percentage                                          0.0
QUANTITY_percentage                                             0.0
LANGUAGE_percentage                                             0.0
Word count                                                       27
Max word length                                                  10
Min word length                                                   1
Average word length                                        4.444444
present_verbs                                                     1
past_verbs                                                        0
adjectives                                                        2
adverbs                                                           2
adpositions                                                       1
pronouns                                                          0
TOs                                                               0
determiners                                                       1
conjunctions                                                      0
dots                                                              1
exclamation                                                       0
questions                                                         0
ampersand                                                         0
capitals                                                          6
digits                                                            2
long_word_freq                                                    0
short_word_freq                                                  19
Name: 68447, dtype: object
```

[ ]:
```python
class Net(nn.Module):
    def __init__(self, params):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(params['input_size'], params['hidden_size'])
        self.fc2 = nn.Linear(params['hidden_size'], params['hidden_size'])
        self.fc3 = nn.Linear(params['hidden_size'], params['num_classes'])

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return torch.log_softmax(x, dim=1)
```

[ ]:
```python
from torch.utils.data import Dataset, DataLoader

class FakeNewsDataset(Dataset):
    def __init__(self, features, target):
        self.features = torch.tensor(features)#, dtype=torch.float32)
        self.target = torch.tensor(target)#, dtype=torch.float32)
```

```
    def __len__(self):
        return len(self.features)

    def __getitem__(self, index):
        return self.features[index], self.target[index]
```

[ ]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder


features_train = train_data.drop('BinaryNumTarget', axis=1).values

# Handle string values using label encoding
label_encoder = LabelEncoder()
# Identify columns with string values (assuming dtype is 'str' or 'object')
string_columns = np.array([np.issubdtype(type(col), np.str_) or np.
 ↪issubdtype(type(col), np.object) for col in features_train[0]])

# Apply label encoding to string columns
for col_index in np.where(string_columns)[0]:
    features_train[:, col_index] = label_encoder.fit_transform(features_train[:
 ↪, col_index].astype(str))

features_train = features_train.astype(np.float32)


# for i in range(len(features_train[0])):
#     print(i, type(features_train[1][i]))

target_train = train_data['BinaryNumTarget'].values
train_torch_ds = FakeNewsDataset(features_train, target_train)
train_loader = DataLoader(train_torch_ds, batch_size=params['batch_size'],␣
 ↪shuffle=True)
```

```
C:\Users\roryb\AppData\Local\Temp\ipykernel_10028\2279553126.py:9:
DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`.
To silence this warning, use `object` by itself. Doing this will not modify any
behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  string_columns = np.array([np.issubdtype(type(col), np.str_) or
np.issubdtype(type(col), np.object) for col in features_train[0]])
```

[ ]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

features_test = test_data.drop('BinaryNumTarget', axis=1).values
```

```
# Handle string values using label encoding
label_encoder = LabelEncoder()
# Identify columns with string values (assuming dtype is 'str' or 'object')
string_columns = np.array([np.issubdtype(type(col), np.str_) or np.
 ↪issubdtype(type(col), np.object) for col in features_test[0]])

# Apply label encoding to string columns
for col_index in np.where(string_columns)[0]:
    features_test[:, col_index] = label_encoder.fit_transform(features_test[:,␣
 ↪col_index].astype(str))

features_test = features_test.astype(np.float32)


# for i in range(len(features_train[0])):
#     print(i, type(features_train[1][i]))

target_test = test_data['BinaryNumTarget'].values
test_torch_ds = FakeNewsDataset(features_test, target_test)
test_loader = DataLoader(test_torch_ds, batch_size=params['batch_size'],␣
 ↪shuffle=True)
```

```
C:\Users\roryb\AppData\Local\Temp\ipykernel_10028\395222176.py:8:
DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`.
To silence this warning, use `object` by itself. Doing this will not modify any
behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  string_columns = np.array([np.issubdtype(type(col), np.str_) or
np.issubdtype(type(col), np.object) for col in features_test[0]])
```

[ ]:
```
for i, e in enumerate(test_loader):
    print(e[0].shape)
    break
```

```
torch.Size([64, 45])
```

[ ]:

[ ]:
```
for i, e in enumerate(train_loader):
    print(i, e[0].shape, e[1].shape)
    break
```

```
0 torch.Size([64, 45]) torch.Size([64])
```

[ ]:
```
model = Net(params)
loss = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=params['learning_rate'])
```

[ ]:
```
total_step = len(train_loader)
for epoch in range(params['num_epochs']):
    for i, (features, labels) in enumerate(train_loader):
        features = features.float()
        labels = labels.long()
        outputs = model(features)
        l = loss(outputs, labels)
        optimizer.zero_grad()
        l.backward()
        optimizer.step()
        if (i+1) % 1000 == 0:
            print ('Epoch [{}/{}], Step [{}/{}], Loss: {:.4f}'.format(epoch+1,␣
 ↪params['num_epochs'], i+1, total_step, l.item()))
```

```
Epoch [1/10], Step [1000/1668], Loss: 3.6069
Epoch [2/10], Step [1000/1668], Loss: 0.2341
Epoch [3/10], Step [1000/1668], Loss: 0.3327
Epoch [4/10], Step [1000/1668], Loss: 0.3476
Epoch [5/10], Step [1000/1668], Loss: 0.3491
Epoch [6/10], Step [1000/1668], Loss: 0.2927
Epoch [7/10], Step [1000/1668], Loss: 0.2368
Epoch [8/10], Step [1000/1668], Loss: 0.3204
Epoch [9/10], Step [1000/1668], Loss: 0.2981
Epoch [10/10], Step [1000/1668], Loss: 0.2441
```

[ ]:
```
# eval model
model.eval()
with torch.no_grad():
    correct = 0
    total = 0
    for features, labels in test_loader:
        features = features.float()
        labels = labels.long()
        outputs = model(features)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    print('Train Accuracy of the model on the {} test tweets: {} %'.
 ↪format(total, np.round(100 * correct / total, 2)))
```

```
Train Accuracy of the model on the 27465 test tweets: 80.87 %
```