fake-news-detect-merge

June 28, 2024

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split as ttp
from sklearn.metrics import classification_report
import re
import string
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

[3]:
```
data_true=pd.read_csv("/content/drive/MyDrive/panda_dataset/True.csv")
data_fake=pd.read_csv("/content/drive/MyDrive/panda_dataset/Fake.csv")
```

[5]:
```
data_true['label'] = 1
data_fake['label'] = 0
```

[6]:
```
data_true_manual_testing = data_true.tail(10)
data_true = data_true.iloc[:-10]
data_fake_manual_testing = data_fake.tail(10)
data_fake = data_fake.iloc[:-10]
```

[7]:
```
data_manual_testing = pd.concat([data_true_manual_testing,␣
  ↪data_fake_manual_testing], axis=0)
data_manual_testing.to_csv('manual_testing.csv', index=False)
```

[8]:
```
data_merge = pd.concat([data_true, data_fake], axis=0)
```

[9]:
```
data = data_merge.drop(['title', 'subject', 'date'], axis=1)
data = data.sample(frac=1).reset_index(drop=True)
```

[10]:
```
def filtering(data):
  text=data.lower()
  text=re.sub('\[.*?\]', '', text)
  text=re.sub("\\W"," ",text)
  text=re.sub('https?://\S+|www\.\S+', '', text)
  text=re.sub('<.*?>+', '', text)
  text=re.sub('[%s]' % re.escape(string.punctuation), '', text)
  text=re.sub('\n', '', text)
  text=re.sub('\w*\d\w*', '', text)
  return text
```

[11]:
```
data['text'] = data['text'].apply(filtering)
```

[12]:
```
vectorizer = TfidfVectorizer()
x = vectorizer.fit_transform(data['text'])
y = data['label']
```

[13]:
```
x_train, x_test, y_train, y_test = ttp(x, y, test_size=0.25, random_state=0)
```

[ ]:
```
LR = LinearRegression()
LR.fit(x, y)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-14-d12cc5b49d9c> in <cell line: 2>()
      1 LR = LinearRegression()
----> 2 LR.fit(x, y)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_base.py in␣
 ↪fit(self, X, y, sample_weight)
    688
    689             if y.ndim < 2:
--> 690                 self.coef_ = lsqr(X_centered, y)[0]
    691             else:
    692                 # sparse_lstsq cannot handle y with shape (M, K)

/usr/local/lib/python3.10/dist-packages/scipy/sparse/linalg/_isolve/lsqr.py in␣
 ↪lsqr(A, b, damp, atol, btol, conlim, iter_lim, show, calc_var, x0)
    432             u = (1/beta) * u
    433             anorm = sqrt(anorm**2 + alfa**2 + beta**2 + dampsq)
--> 434             v = A.rmatvec(u) - beta * v
    435             alfa = np.linalg.norm(v)
    436             if alfa > 0:

/usr/local/lib/python3.10/dist-packages/scipy/sparse/linalg/_interface.py in␣
 ↪rmatvec(self, x)
    279             raise ValueError('dimension mismatch')
    280
--> 281         y = self._rmatvec(x)
    282
    283         if isinstance(x, np.matrix):

/usr/local/lib/python3.10/dist-packages/scipy/sparse/linalg/_interface.py in␣
 ↪_rmatvec(self, x)
    595         if func is None:
    596             raise NotImplementedError("rmatvec is not defined")
--> 597         return self.__rmatvec_impl(x)
    598
    599     def _rmatmat(self, X):

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_base.py in␣
 ↪rmatvec(b)
    681
    682             def rmatvec(b):
--> 683                 return X.T.dot(b) - X_offset_scale * b.
 ↪dot(sample_weight_sqrt)
    684
    685             X_centered = sparse.linalg.LinearOperator(

/usr/local/lib/python3.10/dist-packages/scipy/sparse/_base.py in dot(self, other)
    409             return self * other
    410         else:
--> 411             return self @ other
    412
    413     def power(self, n, dtype=None):

/usr/local/lib/python3.10/dist-packages/scipy/sparse/_base.py in␣
 ↪__matmul__(self, other)
    622             raise ValueError("Scalar operands are not allowed, "
    623                              "use '*' instead")
--> 624         return self._mul_dispatch(other)
    625
    626     def __rmatmul__(self, other):

/usr/local/lib/python3.10/dist-packages/scipy/sparse/_base.py in␣
 ↪_mul_dispatch(self, other)
    520             # Fast path for the most common case
    521             if other.shape == (N,):
--> 522                 return self._mul_vector(other)
    523             elif other.shape == (N, 1):
    524                 return self._mul_vector(other.ravel()).reshape(M, 1)

/usr/local/lib/python3.10/dist-packages/scipy/sparse/_compressed.py in␣
 ↪_mul_vector(self, other)
    486         # csr_matvec or csc_matvec
    487         fn = getattr(_sparsetools, self.format + '_matvec')
--> 488         fn(M, N, self.indptr, self.indices, self.data, other, result)
    489 
    490         return result

KeyboardInterrupt:
```

[ ]:

```
LO = LogisticRegression(max_iter=1000)
LO.fit(x_train, y_train)
```

[ ]:

```
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(x_train, y_train)

DTC = DecisionTreeClassifier()
DTC.fit(x_train, y_train)

RFC = RandomForestClassifier(random_state=0)
RFC.fit(x_train, y_train)
```

[ ]:

```
DT=DecisionTreeClassifier()
DT.fit(xv_train,y_train)
```

[ ]:

```
DecisionTreeClassifier()
```

[ ]:

```
# ... (Your existing code)

# Train models
LR = LinearRegression()
LR.fit(x_train, y_train)

LO = LogisticRegression(max_iter=1000)
LO.fit(x_train, y_train)

k=2
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(x_train, y_train)

DTC = DecisionTreeClassifier()
DTC.fit(x_train, y_train)

RFC = RandomForestClassifier(random_state=0)
RFC.fit(x_train, y_train)

# Make predictions (replace 'your_text' with the text you want to classify)
def predict_news(text, model):
  text_vectorized = vectorizer.transform([filtering(text)])
  prediction = model.predict(text_vectorized)
```

```python
  if prediction == 1:
    return "This news is likely true."
  else:
    return "This news is likely fake."

your_text = "Enter your news text here"
print("Linear Regression:", predict_news(your_text, LR))
print("Logistic Regression:", predict_news(your_text, LO))
print("K-NN:", predict_news(your_text, knn))
print("Decision Tree:", predict_news(your_text, DTC))
print("Random Forest:", predict_news(your_text, RFC))
```

```
Linear Regression: This news is likely fake.
Logistic Regression: This news is likely fake.
K-NN: This news is likely fake.
Decision Tree: This news is likely fake.
Random Forest: This news is likely fake.
```