Statistical Machine Learnings

Author: Alikhan Semembayev

1. Perform necessary data preprocessing, e.g. removing punctuation and stop words, stemming, lemmatizing. You may use the outputs from previous weekly assignments. (10 points)

In [ ]:
```
from collections import defaultdict
import demoji
import svgling
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import pos_tag
from autocorrect import Speller
import re

# Initialize tools
spell = Speller()
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()

email_re = r"\b[A-Za-z]+@\S+\b"
ssn_re = r"\b[0-9]{3}-[0-9]{2}-[0-9]{4}\b"
ip_re = r"\b\d{1,3}[.]\d{1,3}[.]\d{1,3}[.]\d{1,3}\b"

street_number_re = r"^\d{1,}"
street_name_re = r"[a-zA-Z0-9\s]+,?"
city_name_re = r" [a-zA-Z]+(\,)?"
state_abbrev_re = r" [A-Z]{2}"
postal_code_re = r" [0-9]{5}$"
address_pattern_re = r"" + street_number_re + street_name_re + city_name_re + st


def clean_text(text):
    # Replace emojis
    text = demoji.replace(text)

    # Remove smart quotes and dashes
    text = text.replace("“", "\"").replace("”", "\"").replace("-", " ").replace(

    # Lowercase text
    text = text.lower()

    # Tokenize text
    words = word_tokenize(text)
    # print(words)

    # Spelling correction + replace all t with not
    words = ['not' if word == 't' else (
```

```
        'ADDRESS' if re.match(address_pattern_re, word)
        else (
            'EMAIL' if re.match(email_re, word)
            else (
                'SSN' if re.match(ssn_re, word)
                else (
                    'IP' if re.match(ip_re, word)
                    else spell(word)
                )
            )
        )
    ) for word in words]

    # Remove stop words and non-alphabetic tokens and punctuation
    words = [word for word in words if word.isalnum() and word not in stop_words

    # POS tagging and Lemmatization
    tagged_words = pos_tag(words)

    tag_map = defaultdict(lambda: "n")
    tag_map["N"] = "n"
    tag_map["V"] = "v"
    tag_map["J"] = "a"
    tag_map["R"] = "r"

    words = [lemmatizer.lemmatize(word, pos=tag_map[tag[0]]) for word, tag in ta

    # Return cleaned words as a single string
    return ' '.join(words)
```

In [ ]:
```
import pandas as pd

data = (pd.read_csv('../../../../data/text/combined_raw.csv'))
data = data.dropna(how='any')

for row in data.values:
    row[0] = clean_text(row[0])

data.to_csv('../../../../data/text/combined_cleaned.csv', index=False)
```

In [1]:
```
import pandas as pd

data = (pd.read_csv('../../../../data/text/combined_cleaned.csv'))
data = data.dropna(how='any')

print(data.head(10))
```

```
                                                text    emotion
0  freshwater fish drink water skin via osmosis s...      happy
1  think everyone must use daily become grained e...    neutral
2  agree google headquarters mountain view califo...    neutral
3  thats funny current ceo sunday ficha didnt kno...    neutral
4  oh yeah not know either also want go google al...  surprised
5                                              say    surprised
6        yeah apparently lol instead hire people row      happy
7  thats funny guess imaginative leave huge tech ...  surprised
8  yeah exactly sure cheap one thing bet not expl...  surprised
9  remember hearing immortality waste jellyfish h...    neutral
```

2. Propose a binary classification problem from your project data and identify the columns that you will use to solve the problem. You may need to create new columns of data. (20 points)

In [4]:
```
import pandas as pd

data = (pd.read_csv('../../../../data/text/combined_cleaned.csv'))
data = data.dropna(how='any')
```

In [5]:
```
emotions = ['happy', 'surprised', 'neutral', 'sad', 'fear', 'angry', 'disgust']

for emotion in emotions:
    data[f'is_{emotion}'] = data['emotion'].apply(lambda x: 1 if x == emotion el

data = data.drop(columns=['emotion'])

data.to_csv('../../../../data/text/combined_cleaned_multilabel.csv', index=False
data.head()
```

Out[5]:

	text	is_happy	is_surprised	is_neutral	is_sad	is_fear	is_angry	is_disgust
0	freshwater fish drink water skin via osmosis s...	1	0	0	0	0	0	0
1	think everyone must use daily become grained e...	0	0	1	0	0	0	0
2	agree google headquarters mountain view califo...	0	0	1	0	0	0	0
3	thats funny current ceo sunday ficha didnt kno...	0	0	1	0	0	0	0
4	oh yeah not know either also want go google al...	0	1	0	0	0	0	0

3. Compute TF-IDF vectors on the text data. (10 points)

In [6]:
```
import pandas as pd

data = (pd.read_csv('../../../../data/text/combined_cleaned_multilabel.csv'))
data = data.dropna(how='any')
```

In [7]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

X_tfidf = vectorizer.fit_transform(data['text'])

print(X_tfidf.shape)
```

```
(147380, 32085)
```

4. Solve your binary classification problem with the Naïve Bayes classifier. (30 points)

In [9]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Define features and labels
X = X_tfidf
y = data['is_happy']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Initialize and train the Naive Bayes classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred_nb = nb_classifier.predict(X_test)

# Evaluate the classifier
print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Accuracy: 0.7338512688288777
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.98      0.84     20896
           1       0.72      0.14      0.23      8580

    accuracy                           0.73     29476
   macro avg       0.73      0.56      0.54     29476
weighted avg       0.73      0.73      0.66     29476
```

5. Solve your binary classification problem with the SVC classifier. (30 points)

In [11]:
```
from sklearn.svm import SVC

# Initialize and train the SVC classifier
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred_svc = svc_classifier.predict(X_test)

# Evaluate the classifier
print("SVC Accuracy:", accuracy_score(y_test, y_pred_svc))
print("SVC Classification Report:")
print(classification_report(y_test, y_pred_svc))
```

```
SVC Accuracy: 0.7476591124983037
SVC Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.96      0.84     20896
           1       0.71      0.23      0.34      8580

    accuracy                           0.75     29476
   macro avg       0.73      0.59      0.59     29476
weighted avg       0.74      0.75      0.70     29476
```

In [12]:
```
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, y_pred_svc, normalize=None)
print(conf_matrix)
```

```
[[20089   807]
 [ 6631  1949]]
```