## Loading data

In [1]:

```
def split_in_sets(data):
    essay_sets = []
    min_scores = []
    max_scores = []
    for s in range(1,9):
        essay_set = data[data["essay_set"] == s]
        essay_set.dropna(axis=1, inplace=True)
        n, d = essay_set.shape
        set_scores = essay_set["domain1_score"]
        print ("Set", s, ": Essays = ", n , "\t Attributes = ", d)
        min_scores.append(set_scores.min())
        max_scores.append(set_scores.max())
        essay_sets.append(essay_set)
    return (essay_sets, min_scores, max_scores)
```

In [96]:

```
dataset_path = "./asap-aes/training_set_rel3.tsv"

import os
import pandas as pd

data = pd.read_csv(dataset_path, sep="\t", encoding="ISO-8859-1")
min_scores = [2, 1, 0, 0, 0, 0, 0, 0]
max_scores = [12, 6, 3, 3, 4, 4, 30, 60]

essay_sets, data_min_scores, data_max_scores = split_in_sets(data)
set1, set2, set3, set4, set5, set6, set7, set8 = tuple(essay_sets)
data.dropna(axis=1, inplace=True)

data.drop(columns=["rater1_domain1", "rater2_domain1"], inplace=Tru
print("All Data:", len(data))
data.head()
```

```
Set 1 : Essays =  1783   Attributes =  6
Set 2 : Essays =  1800   Attributes =  9
Set 3 : Essays =  1726   Attributes =  6
Set 4 : Essays =  1770   Attributes =  6
Set 5 : Essays =  1805   Attributes =  6
Set 6 : Essays =  1800   Attributes =  6
Set 7 : Essays =  1569   Attributes =  14
Set 8 : Essays =  723    Attributes =  18
All Data: 12976
```

Out[96]:

| | essay_id | essay_set | essay | domain1_score |
|---|---|---|---|---|
| 0 | 1 | 1 | Dear local newspaper, I think effects computer... | 8 |
| 1 | 2 | 1 | Dear @CAPS1 @CAPS2, I believe that using compu... | 9 |
| 2 | 3 | 1 | Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl... | 7 |
| 3 | 4 | 1 | Dear Local Newspaper, @CAPS1 I have found that... | 10 |
| 4 | 5 | 1 | Dear @LOCATION1, I know having computers has a... | 8 |

In [3]:

```
print("Minimum Scores: ", min_scores)
print("Maximum Scores: ", max_scores)
```

```
Minimum Scores:  [2, 1, 0, 0, 0, 0, 0, 0]
Maximum Scores:  [12, 6, 3, 3, 4, 4, 30, 60]
```

In [4]:

```
#Dataset keys
essay_id_key = "essay_id"
essay_set_key = "essay_set"
essay_key = "essay"
domain1_score_key = "domain1_score"
```

In [85]:

```
#Feature keys
feature_keys = {
    "char_count_key": "char_count",
    "word_count_key": "word_count",
    "diff_words_count_key": "diff_words_count",
    "word_count_root_key": "word_count_root",
    "sen_count_key": "sen_count",
    "avg_word_len_key": "avg_word_len",
    "avg_sen_len_key": "avg_sen_len",
    "l5_word_count_key": "l5_word_count",
    "l6_word_count_key": "l6_word_count",
    "l7_word_count_key": "l7_word_count",
    "l8_word_count_key": "l8_word_count",
}
```

In [86]:

```
#Extra features
extra_feature_keys = {
    # "grammer_error_count_key": "grammer_error_count",
    "spelling_error_count_key": "spelling_error_count",
    "stopwords_count_key": "stopwords_count",
    "small_sentences_count_key": "small_sentence_count", #sentences
    # "beautiful_words_count_key": "beautiful_words_count",
    "punctuations_count_key": "punctuations_count",
    "verbs_count_key": "verbs_count",
    "adverbs_count_key": "adverbs_count",
    "nouns_count_key": "nouns_count",
    "adjectives_count_key": "adjective_count",
}
```

In [87]:

```
feature_keys_list = list(feature_keys.values())
extra_feature_keys_list = list(extra_feature_keys.values())
all_feature_keys_list = feature_keys_list + extra_feature_keys_list

print("Basic 11 features: ", feature_keys_list, "\n")
print("Extra features: ", extra_feature_keys_list, "\n")
print("All features: ", all_feature_keys_list, "\n")
```

```
Basic 11 features:  ['char_count', 'word_count', 'diff_words_count
', 'word_count_root', 'sen_count', 'avg_word_len', 'avg_sen_len',
'l5_word_count', 'l6_word_count', 'l7_word_count', 'l8_word_count'
]

Extra features:  ['spelling_error_count', 'stopwords_count', 'smal
l_sentence_count', 'punctuations_count', 'verbs_count', 'adverbs_c
ount', 'nouns_count', 'adjective_count']

All features:  ['char_count', 'word_count', 'diff_words_count', 'w
ord_count_root', 'sen_count', 'avg_word_len', 'avg_sen_len', 'l5_w
ord_count', 'l6_word_count', 'l7_word_count', 'l8_word_count', 'sp
elling_error_count', 'stopwords_count', 'small_sentence_count', 'p
unctuations_count', 'verbs_count', 'adverbs_count', 'nouns_count',
'adjective_count']
```

In [9]:

```
from spellchecker import SpellChecker
spell = SpellChecker()

misspelled = spell.unknown(['something', 'is', 'hapenning', 'here']
for word in misspelled:
    print(word)
```

```
hapenning
```

In [102]:

```
import numpy as np
import nltk
import re
from nltk.corpus import stopwords
import language_check
from spellchecker import SpellChecker
from collections import Counter
from nltk.tag.perceptron import PerceptronTagger

tagger=PerceptronTagger() # load outside
tool = language_check.LanguageTool('en-US')
spell = SpellChecker()
spell.word_frequency.load_words(["PERSON", "ORGANIZATION", "LOCATIO

def sentence_to_word_list(sentence, remove_stopwords):
    # Remove non letter from sentenece and stop words
    sen_char_count = 0
    sen_word_count = 0
    sen_word_count = 0
    l5_sen_word_count = 0
    l6_sen_word_count = 0
    l7_sen_word_count = 0
    l8_sen_word_count = 0
    sen_diff_words = set()
    ### Extra Features ###
    sen_verbs_count = 0
    sen_adverbs_count = 0
    sen_nouns_count = 0
    sen_adjectives_count = 0
    sen_spelling_error_count = 0
    sen_stopwords_count = 0
    is_small_sentence = 0

    stops = set(stopwords.words("english"))
    all_words = sentence.lower().split()

    # count= Counter([j for i,j in tagger.tag(all_words)])
    # sen_verbs_count = count['VB'] + count['VBG'] + count['VBP'] +
    # sen_adverbs_count = count['RB'] + count['RBR'] + count['RBS']
    # sen_nouns_count = count['NN'] + count['NNS'] + count['NNPS']
    # sen_adjectives_count = count['JJ'] + count['JJR']

    kept_words = []


    if len(all_words) <= 4: is_small_sentence = 1


    misspelled = spell.unknown(all_words)
    sen_spelling_error_count = len(misspelled)

    for word in all_words:
        sen_char_count += len(word)
        sen_word_count += 1
        word_len = len(word)
        if word_len > 5:
            l5_sen_word_count += 1
        if word_len > 6:
            l6_sen_word_count += 1
        if word_len > 7:
            l7_sen_word_count += 1
        if word_len > 8:
            l8_sen_word_count += 1

        sen_diff_words.add(word)

        isStopword = word in stops
        if isStopword: sen_stopwords_count += 1

        if remove_stopwords and not isStopword:
            kept_words.append(word)
        else:
            kept_words.append(word)
            kept_words.append(word)

    features = {
         feature_keys["char_count_key"]: sen_char_count,
         feature_keys["word_count_key"]: sen_word_count,
         feature_keys["l5_word_count_key"]: l5_sen_word_count,
         feature_keys["l6_word_count_key"]: l6_sen_word_count,
         feature_keys["l7_word_count_key"]: l7_sen_word_count,
         feature_keys["l8_word_count_key"]: l8_sen_word_count,
         feature_keys["diff_words_count_key"]: sen_diff_words
    }

    extra_features = {
        extra_feature_keys["small_sentences_count_key"]: is_small_s
        extra_feature_keys["spelling_error_count_key"]: sen_spellin
        extra_feature_keys["stopwords_count_key"]: sen_stopwords_co
        extra_feature_keys["verbs_count_key"]: sen_verbs_count,
        extra_feature_keys["adverbs_count_key"]: sen_adverbs_count,
        extra_feature_keys["nouns_count_key"]: sen_nouns_count,
        extra_feature_keys["adjectives_count_key"]: sen_adjectives_
    }

    return (kept_words, features, extra_features)

def essay_to_sentences(essay, remove_stopwords = False):
    # Convert essay into sentence


    tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
    sentences = tokenizer.tokenize(essay.strip())
    split_sentences = []

    char_count = 0
    word_count = 0
    diff_words = set()
    word_count_root = 0
    sen_count = 0
    avg_word_len = 0
    avg_sen_len = 0
    l5_word_count = 0
    l6_word_count = 0
    l7_word_count = 0
    l8_word_count = 0


    ### Extra Features ###
    spelling_error_count = 0
    stopwords_count = 0
    small_sentences_count = 0
    punctuation_count = 0
    grammer_error_count = 0
    small_sentences_count = 0
    verbs_count = 0
    adverbs_count = 0
    nouns_count = 0
    nouns_count = 0
    adjectives_count = 0

    all_words = nltk.word_tokenize(essay)
    count= Counter([j for i,j in tagger.tag(all_words)])
    verbs_count = count['VB'] + count['VBG'] + count['VBP'] + count
    adverbs_count = count['RB'] + count['RBR'] + count['RBS']
    nouns_count = count['NN'] + count['NNS'] + count['NNPS'] + coun
    adjectives_count = count['JJ'] + count['JJR']

    punctuation = ['.','?', '!', ':', ';']
    for punct in punctuation:
        punctuation_count += essay.count(punct)

    for sentence in sentences:
        if len(sentence) > 0:
            sentence = re.sub("[^a-zA-Z]", " ", sentence)
            # grammer_error_count += len(tool.check(sentence))


            kept_words, features, extra_features = sentence_to_word
            split_sentences.append(kept_words)

            sen_count +=1
            char_count += features[feature_keys["char_count_key"]]
            word_count += features[feature_keys["word_count_key"]]
            l5_word_count += features[feature_keys["l5_word_count_k
            l6_word_count += features[feature_keys["l6_word_count_k
            l7_word_count += features[feature_keys["l7_word_count_k
            l8_word_count += features[feature_keys["l8_word_count_k
            diff_words = diff_words|features[feature_keys["diff_wor
            ### Extra Features ###
            spelling_error_count += extra_features[extra_feature_ke
            stopwords_count += extra_features[extra_feature_keys["s
            small_sentences_count += extra_features[extra_feature_k
            # verbs_count += extra_features[extra_feature_keys["ver
            # adverbs_count += extra_features[extra_feature_keys["a
            # nouns_count += extra_features[extra_feature_keys["nou
            # adjectives_count += extra_features[extra_feature_keys


    word_count_root = word_count ** (1/4)
    avg_word_len = char_count / word_count
    avg_sen_len = word_count / sen_count

    features = {
        feature_keys["char_count_key"]: char_count,
        feature_keys["word_count_key"]: word_count,
        feature_keys["diff_words_count_key"]: len(diff_words),
        feature_keys["word_count_root_key"]: word_count_root,
        feature_keys["sen_count_key"]: sen_count,
        feature_keys["avg_word_len_key"]: avg_word_len,
        feature_keys["avg_sen_len_key"]: avg_sen_len,
        feature_keys["l5_word_count_key"]: l5_word_count,
        feature_keys["l6_word_count_key"]: l6_word_count,
        feature_keys["l6_word_count_key"]: l6_word_count,
        feature_keys["l7_word_count_key"]: l7_word_count,
        feature_keys["l8_word_count_key"]: l8_word_count
    }

    extra_features = {
        # extra_feature_keys["grammer_error_count_key"]: grammer_er
        extra_feature_keys["spelling_error_count_key"]: spelling_er
        extra_feature_keys["stopwords_count_key"]: stopwords_count,
        extra_feature_keys["small_sentences_count_key"]: small_sent
        extra_feature_keys["punctuations_count_key"]: punctuation_c
        extra_feature_keys["verbs_count_key"]: verbs_count,
        extra_feature_keys["adverbs_count_key"]: adverbs_count,
        extra_feature_keys["nouns_count_key"]: nouns_count,
        extra_feature_keys["adjectives_count_key"]: adjectives_coun
    }

    return (split_sentences, features, extra_features)
```

In [78]:

```
import pprint
from time import time

pp = pprint.PrettyPrinter(indent=4)

#Featrues
first_essay = data.iloc[0][essay_key]
print(first_essay)
start = time()
split_sentences, features, extra_features = essay_to_sentences(firs
end = time()
print("Execution time:", end-start)
# print(split_sentences)
print("\n\nFeatures: ")
pp.pprint(features)

print("\n\n Extra Features: ")
pp.pprint(extra_features)
```

```
Dear local newspaper, I think effects computers have on people are
great learning skills/affects because they give us time to chat wi
th friends/new people, helps us learn about the globe(astronomy) a
nd keeps us out of troble! Thing about! Dont you think so? How wou
ld you feel if your teenager is always on the phone with friends!
Do you ever time to chat with your friends or buisness partner abo
ut things. Well now - there's a new way to chat the computer, thei
rs plenty of sites on the internet to do so: @ORGANIZATION1, @ORGA
NIZATION2, @CAPS1, facebook, myspace ect. Just think now while you
r setting up meeting with your boss on the computer, your teenager
is having fun on the phone not rushing to get off cause you want t
o use it. How did you learn about other countrys/states outside of
yours? Well I have by computer/internet, it's a new way to learn a
bout what going on in our time! You might think your child spends
```

```
a lot of time on the computer, but ask them so question about the
economy, sea floor spreading or even about the @DATE1's you'll be
surprise at how much he/she knows. Believe it or not the computer
is much interesting then in class all day reading out of books. If
your child is home on your computer or at a local library, it's be
tter than being out with friends being fresh, or being perpressure
d to doing something they know isnt right. You might not know wher
e your child is, @CAPS2 forbidde in a hospital bed because of a dr
ive-by. Rather than your child on the computer learning, chatting
or just playing games, safe and sound in your home or community pl
ace. Now I hope you have reached a point to understand and agree w
ith me, because computers can have great effects on you or child b
ecause it gives us time to chat with friends/new people, helps us
learn about the globe and believe or not keeps us out of troble. T
hank you for listening.
Execution time: 0.04050278663635254


Features:
{   'avg_sen_len': 21.875,
    'avg_word_len': 4.222857142857142,
    'char_count': 1478,
    'diff_words_count': 164,
    'l5_word_count': 74,
    'l6_word_count': 59,
    'l7_word_count': 34,
    'l8_word_count': 13,
    'sen_count': 16,
    'word_count': 350,
    'word_count_root': 4.3253077270721105}


 Extra Features:
{   'adjective_count': 22,
    'adverbs_count': 21,
    'nouns_count': 84,
    'punctuations_count': 17,
    'small_sentence_count': 3,
    'spelling_error_count': 9,
    'stopwords_count': 184,
    'verbs_count': 67}
```

In [100]:

```python
def makeDataFrame(data):
    all_features = {}
    all_scores = {}


    for row in range(len(data)):
        # if row % 500 == 0: print("Processed ", row, " essays of",
        essay_data = data.iloc[row]
        essay = essay_data[essay_key]
        essay_id = essay_data[essay_id_key]
        essay_score = essay_data[domain1_score_key]
        _, features, extra_features = essay_to_sentences(essay)

        combined_features = {}
        combined_features.update(features)
        combined_features.update(extra_features)
        all_features[essay_id] = combined_features
        all_scores[essay_id] = essay_score

    X = pd.DataFrame.from_dict(all_features, orient="index")
    y = pd.DataFrame.from_dict(all_scores, orient="index")

    return(X, y)
```

In [80]:

```
from time import time
start = time()
X, y = makeDataFrame(data)
end = time()
print("Execution time to make dataframe: ", (end-start))
```

```
Processed  0  essays of 12976  rows.
Processed  500  essays of 12976  rows.
Processed  1000  essays of 12976  rows.
Processed  1500  essays of 12976  rows.
Processed  2000  essays of 12976  rows.
Processed  2500  essays of 12976  rows.
Processed  3000  essays of 12976  rows.
Processed  3500  essays of 12976  rows.
Processed  4000  essays of 12976  rows.
Processed  4500  essays of 12976  rows.
Processed  5000  essays of 12976  rows.
Processed  5500  essays of 12976  rows.
Processed  6000  essays of 12976  rows.
Processed  6500  essays of 12976  rows.
Processed  7000  essays of 12976  rows.
Processed  7500  essays of 12976  rows.
Processed  8000  essays of 12976  rows.
Processed  8500  essays of 12976  rows.
Processed  9000  essays of 12976  rows.
Processed  9500  essays of 12976  rows.
Processed  10000  essays of 12976  rows.
Processed  10500  essays of 12976  rows.
Processed  11000  essays of 12976  rows.
Processed  11500  essays of 12976  rows.
Processed  12000  essays of 12976  rows.
Processed  12500  essays of 12976  rows.
Execution time to make dataframe:  269.11509823799133
```

In [97]:

```
# Features for essay
print(len(X))
X.head()
```

```
12976
```

Out[97]:

| | char_count | word_count | diff_words_count | word_count_root | sen_count | avg_word_len |
|---|---|---|---|---|---|---|
| 1 | 1478 | 350 | 164 | 4.325308 | 16 | 4.222857 |
| 2 | 1814 | 423 | 192 | 4.535081 | 20 | 4.288416 |
| 3 | 1222 | 283 | 147 | 4.101537 | 14 | 4.318021 |
| 4 | 2510 | 530 | 232 | 4.798096 | 27 | 4.735849 |
| 5 | 2046 | 473 | 200 | 4.663535 | 30 | 4.325581 |

In [82]:

```
# Score labels
y.head()
```

Out[82]:

|   | 0 |
|---|---|
| 1 | 8 |
| 2 | 9 |
| 3 | 7 |
| 4 | 10 |
| 5 | 8 |

In [98]:

```
X_basic = X[feature_keys_list]
X_basic.head()
print(len(X_basic))
```

```
12976
```

In [91]:

```
# Training on features using linear regression
from time import time
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import cohen_kappa_score as kappa


def evaluate(X, y):

    model = LinearRegression()

    #Simple K-Fold cross validation. 5 folds.
    kf = KFold(n_splits=5, shuffle=True)
    cv = kf.split(X)
    results = []

    for traincv, testcv in cv:
            X_test, X_train, y_test, y_train = X.iloc[testcv], X.il

            model.fit(X_train,y_train)
            start = time()
            y_pred = model.predict(X_test)
            y_pred = [item for sublist in y_pred for item in sublis
            y_pred = np.around(y_pred, decimals=0).astype(int)
            y_test = [item for sublist in y_test.values for item in

            end = time()

            result = kappa(y_test,y_pred,labels=None, weights='quad
            results.append(result)

    return (np.array(results).mean())
```

In [111]:

```
# Using all data set
np.random.seed(1)
print("Evaluation using basic 11 features")
k = evaluate(X_basic, y)
print("Kappa Score (all essays):", k)
```

```
Evaluation using basic 11 features
Kappa Score (all essays): 0.7451946428948097
```

In [112]:

```
np.random.seed(1)
print("Evaluation using all features")
k = evaluate(X, y)
print("Kappa Score (all essays):", k)
```

```
Evaluation using all features
Kappa Score (all essays): 0.8177840358251901
```

In [105]:
```
X_set_list = []
y_set_list = []
essay_sets = [set1, set2, set3, set4, set5, set6, set7, set8]
for set_no in range(8):
    X_set, y_set = makeDataFrame(essay_sets[set_no])
        X_set_list.append(X_set)
            y_set_list.append(y_set) = makeDataFrame(essay_sets[set
    X_set_list.append(X_set)
    y_set_list.append(y_set)
```

In [114]:
```
# Training on individual dataset
np.random.seed(1)
print("Training sets on basic 11 features")
for set_no in range(8):
    X_basic_set, y_set = X_set_list[set_no][feature_keys_list], y_s

    print("Kappa Score for Set", (set_no+1), ":", evaluate(X_basic_
```

```
Training sets on basic 11 features
Kappa Score for Set 1 : 0.8331024981343134
Kappa Score for Set 2 : 0.6873643522701277
Kappa Score for Set 3 : 0.6521253626079226
Kappa Score for Set 4 : 0.6877399928079586
Kappa Score for Set 5 : 0.7846507383076862
Kappa Score for Set 6 : 0.6736451029505887
Kappa Score for Set 7 : 0.7312752459645284
Kappa Score for Set 8 : 0.71497255413149
```

In [115]:
```
np.random.seed(1)
print("Training sets on all features")
for set_no in range(8):
    X_set, y_set = X_set_list[set_no], y_set_list[set_no]
    print("Kappa Score for Set", (set_no+1), ":", evaluate(X_set, y
```

```
Training sets on all features
Kappa Score for Set 1 : 0.8376629910072593
Kappa Score for Set 2 : 0.6966326483543922
Kappa Score for Set 3 : 0.6474088562447882
Kappa Score for Set 4 : 0.6905195626681292
Kappa Score for Set 5 : 0.7828540979457402
Kappa Score for Set 6 : 0.6869831202604875
Kappa Score for Set 7 : 0.7731733283327601
Kappa Score for Set 8 : 0.7178710463754013
```

In [117]:
```
print("Mean of features:")
X.mean(axis=0)
```

```
Mean of features:
```

Out[117]:
```
char_count               952.516954
word_count               225.458154
diff_words_count         108.591862
word_count_root            3.666353
sen_count                 12.716091
avg_word_len               4.236917
avg_sen_len               20.565670
l5_word_count             53.798166
l6_word_count             35.832306
l7_word_count             21.270962
l8_word_count             11.555256
spelling_error_count       4.819744
stopwords_count          118.090244
small_sentence_count       0.437962
punctuations_count        13.717787
verbs_count               36.444436
adverbs_count             13.360897
nouns_count               56.080996
adjective_count           15.469097
dtype: float64
```

In [119]:
```
print("Deviation of features:")
X.std(axis=0)
```

```
Deviation of features:
```

Out[119]:
```
char_count               745.525805
word_count               178.681107
diff_words_count          64.705354
word_count_root            0.726777
sen_count                 11.151927
avg_word_len               0.355270
avg_sen_len               12.388588
l5_word_count             43.060338
l6_word_count             29.229100
l7_word_count             19.037364
l8_word_count             11.194151
spelling_error_count       4.807119
stopwords_count           95.585648
small_sentence_count       1.364738
punctuations_count        13.106078
verbs_count               32.863780
adverbs_count             13.279126
nouns_count               44.943184
adjective_count           13.754006
dtype: float64
```

In [ ]: