Stress Classification with Dreaddit

Comparing Logistic Regression, Decision Trees, and Naive Bayesian Classifiers

Elliott Chen, COMP 4448

Research Question

The goal of this project is to compare how logistic regression, decision tree based, and naive Bayesian classification models perform when predicting whether a person is stressed.

Data source

This Dreaddit dataset was sourced from Kaggle. It was originally built by Elsbeth Turcan and Kathy McKeown of the Columbia University Department of Computer Science in their 2019 paper *Dreaddit: A Reddit Dataset for Stress Analysis in Social Media.* The dataset revolves around personal Reddit posts submitted to various subreddits and whether the original poster (OP) was determined to be stressed.

The dataset can be found at https://www.kaggle.com/ruchi798/stress-analysis-in-social-media

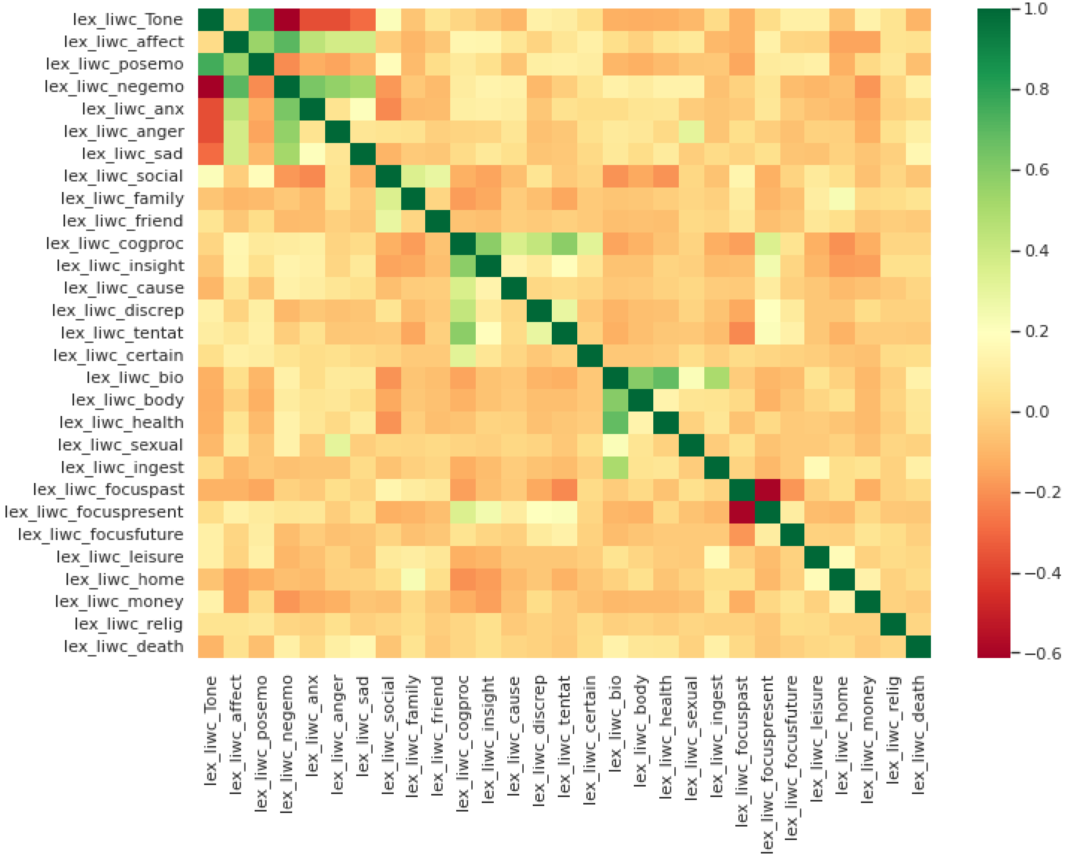
The paper itself can be found at https://aclanthology.org/D19-6213/

Dataset

- The initial dataset consists of 116 columns in 2838 training instances and 715 testing instances, with a variety of categorical and continuous features. The outcome variable, named label, is a binary.
- Most columns were generated by Turcan and McKeown using Linguistic Inquiry and Word Count (LIWC), a commonly used linguistics software tool. Some of these columns track syntax and grammatical information such as word count, words per sentence, and proportion of pronouns, verbs, adjectives, etc. Others track the proportion of words in a given body of text that relate to certain themes, such as anger, health, and leisure.
- Other features include the raw post text and metadata related to Reddit such as upvote count, karma, and post ID.

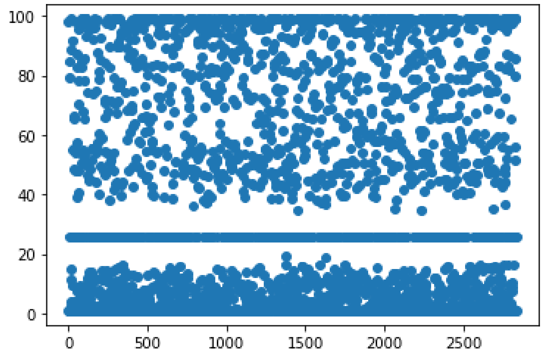
Data Preprocessing

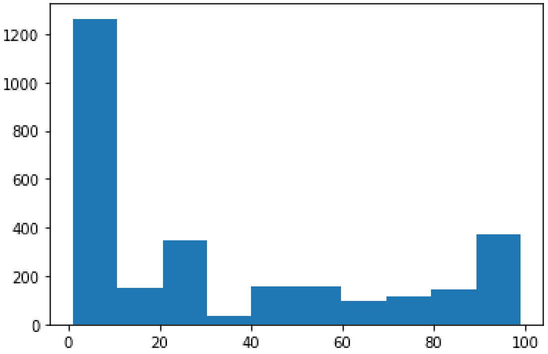
- We use only the LIWC columns relating to personal emotions or physical factors that could affect one's stress levels for logistic regression and decision trees. Inspecting their scatterplots, we find that they are mostly not distributed normally and it does not appear that outliers will be a significant concern. Thus, we use the min-max scaler to standardize them.
- Upon checking for multicollinearity before using logistic regression, we find that a number of our features (tone, positive/negative emotions, bio, and health) are heavily correlated. We drop these columns. This leaves us with 26 continuous features.
- We will only take the raw post text into consideration for our Naive Bayesian analysis. We will have to clean the text data first with NLTK.



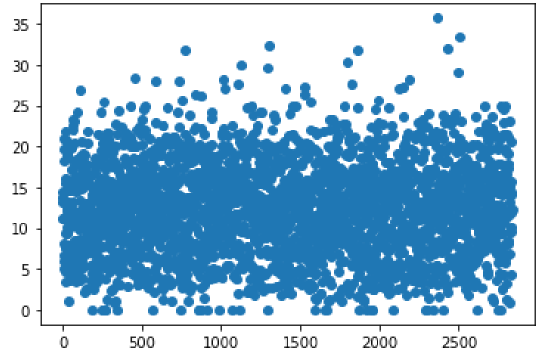
Data Visualizations

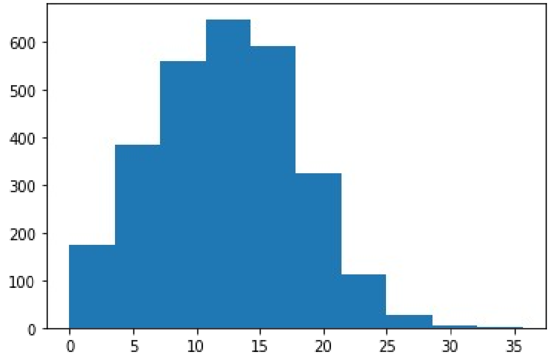
Tone



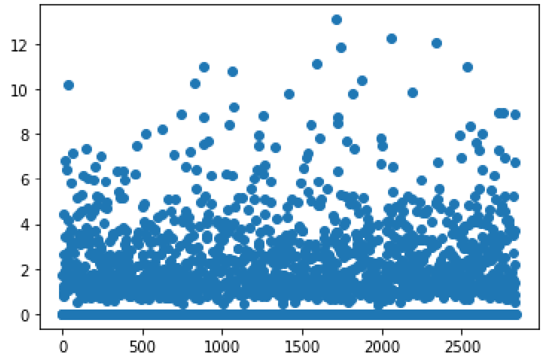


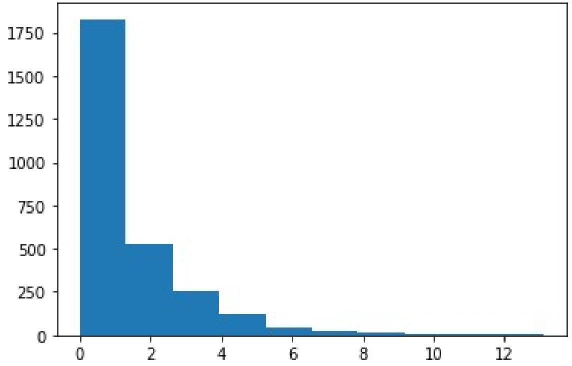
FocusPresent





Health





Statistics

As there are 26 continuous variables, there is not enough time to get into detailed descriptive statistics for each one here. After scaling, however, each ranges from 0 to 1, and all have means less than or equal to 0.33; a majority of the means are less than 0.10.

Data partitioning

As luck would have it, the data from Kaggle comes pre-split into training and testing CSV files. All that remains is to drop the same columns and clean the raw text in our testing set the same way as we did for our training data.

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, recall_score

reg = LogisticRegression()
reg.fit(train_liwc_X, train_y)
train_lreg_pred = reg.predict(train_liwc_X)
test_lreg_pred = reg.predict(test_liwc_X)

print('Overall accuracy')
print(accuracy_score(train_y, train_lreg_pred))
print(accuracy_score(test_y, test_lreg_pred))

print('Recall')
print(recall_score(train_y, train_lreg_pred))
print(recall_score(test_y, test_lreg_pred))
```

```
Overall accuracy
0.7360817477096547
0.7426573426573426
Recall
0.7762096774193549
0.6964769647696477
```

```python
from sklearn.ensemble import RandomForestClassifier

rf_param_grid = {
    'max_depth': [1, 5, 10, 15, dtc.tree_.max_depth],
    'max_features': [0.2, 0.4, 0.6, 0.8, None],
    'n_estimators': [25, 50, 75, 100]
}

gscv = GridSearchCV(RandomForestClassifier(), rf_param_grid, cv=5)
gscv.fit(train_liwc_X, train_y)
best_rf = gscv.best_estimator_
print(best_rf)

train_rf_pred = best_rf.predict(train_liwc_X)
test_rf_pred = best_rf.predict(test_liwc_X)

print('Overall accuracy')
print(accuracy_score(train_y, train_rf_pred))
print(accuracy_score(test_y, test_rf_pred))

print('Recall')
print(recall_score(train_y, train_rf_pred))
print(recall_score(test_y, test_rf_pred))
```

```
RandomForestClassifier(max_depth=15, max_features=0.4, n_estimators=75)
Overall accuracy
0.9975334742776604
0.5160839160839161
Recall
0.9973118279569892
0.8861788617886179
```

```python
from sklearn.model_selection import GridSearchCV

param_grid = {
    'max_depth': range(1, dtc.tree_.max_depth),
    'max_features': [0.2, 0.4, 0.6, 0.8, None]
}

gscv = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5)
gscv.fit(train_liwc_X, train_y)
best_dt = gscv.best_estimator_
print(best_dt)

train_dt_pred = best_dt.predict(train_liwc_X)
test_dt_pred = best_dt.predict(test_liwc_X)

print('Overall accuracy')
print(accuracy_score(train_y, train_dt_pred))
print(accuracy_score(test_y, test_dt_pred))

print('Recall')
print(recall_score(train_y, train_dt_pred))
print(recall_score(test_y, test_dt_pred))
```

```
DecisionTreeClassifier(max_depth=4, max_features=0.4)
Overall accuracy
0.7339675828047921
0.4825174825174825
Recall
0.7634408602150538
0.024390243902439025
```

```python
from sklearn.pipeline import Pipeline

pipe = Pipeline([
    ("tfidf",TfidfVectorizer(stop_words="english")),
    ("nb", MultinomialNB())
])

param_grid = [{
    'tfidf__min_df': [1, 2, 5, 10, 25],
    'tfidf__max_df': [0.1, 0.2, 0.3, 0.4, 0.5],
    'tfidf__max_features': [5, 10, None],
    'tfidf__ngram_range': [(1, 1), (1, 2)],
}]

clf = GridSearchCV(pipe, param_grid)
clf.fit(train_NB_X.text, train_NB_y)
best_nb = clf.best_estimator_
```

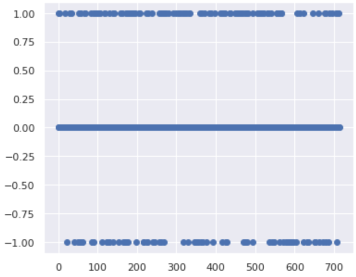
```python
train_nb_pred = best_nb.predict(train_NB_X.text)
test_nb_pred = best_nb.predict(test_NB_X.text)

print('Overall accuracy')
print(accuracy_score(train_NB_y, train_nb_pred))
print(accuracy_score(test_NB_y, test_nb_pred))

print('Recall')
print(recall_score(train_NB_y, train_nb_pred))
print(recall_score(test_NB_y, test_nb_pred))
```

```
Overall accuracy
0.8676314860571832
0.7062937062937062
Recall
0.9367429340511441
0.8617886178861789
```

Evaluation

We should check the residuals of our logistic regressor to make sure they are independent, and are satisfied that they are so.

```
resids = np.array(test_y) - np.array(test_lreg_pred)
sns.set(rc={'figure.figsize':(6, 5)})
plt.scatter(range(715), resids)
plt.show()
```



If the goal of the model is to identify unhealthy signs of stress in a person, false negatives are probably more expensive than false positives if they preclude necessary intervention. We therefore use recall as a metric to take into consideration. Most of our models suffered from overfitting to some degree, with the exception of our logistic regressor, which achieved an overall accuracy of 74% and recall of 70% on the testing set. Our Bayesian classifier performed comparably, with a slightly lower accuracy of 71% but a significantly higher recall of 86%.

Conclusions

Our best models were easily the logistic regression and naive Bayesian classifiers. The regressor was the only model that did not overfit, while the Bayesian model achieved the best balance of accuracy and recall. The decision tree based classifiers, in comparison, suffered from massive overfitting—the single tree’s testing accuracy plummeted to less than 50%, even when tree depth was limited to 4 after hyperparameter tuning. If we were to revisit the decision tree and random forest, we might try a different set of features, but the results here were not promising.

Sources

- https://www.researchgate.net/publication/246699633_Linguistic_inquiry_and_word_count_LIWC
- https://www.kaggle.com/ruchi798/stress-analysis-in-social-media
- https://aclanthology.org/D19-6213/
- http://liwc.wpengine.com/interpreting-liwc-output/