Project_P3

February 3, 2026

0.1 Part a: Model Selection

[2]:
```
## Step 1: Import pandas, read and load data set
import pandas as pd
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold, train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris, load_breast_cancer
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
df = pd.read_csv('project_data.csv')
df
```

[2]:
```
       Unnamed: 0                Tweet  Followers  Friends  Num_tweets  \
0               0  1441497018807906305       1198    605.0        6166
1               1  1431812786099613699        608   1259.0        1811
2               2  1426644707313135617        173    167.0        4306
3               3  1431814908425998337       2540    222.0        6008
4               4  1432862687533441027       4439     11.0        9985
…             …                    …        …      …          …
28818       28815  1431029315274162181      16182   1150.0       58245
28819       28816  1437970083087851520     688077    636.0       12308
28820       28817  1428110541093052418      57068   1225.0       25131
28821       28818  1430722665514377219         66    121.0         549
28822       28819  1441165693974503442        169    276.0        2910

       Verified  Listed_count Location  Age  Length  Num_users  \
0         False             1     True    7       6          4
1         False             5     True    2       2          2
2         False             0     True    0       4          2
3         False             0     True    9       2          2
```

```
4        False          55    True   2      4        3
…        …          …      …  …     …       …
28818     True         685   False  12      2        2
28819     True         361   False  10    247      218
28820    False           0    True   5     22       21
28821    False           3    True   1      4        3
28822    False           7    True   0      2        2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2    0.235235                       0.0
1                       0    0.085582                       0.0
2                       2    0.076877                       0.0
3                       0    0.095684                       0.0
4                       1    0.165919                       0.0
…                   …           …                       …
28818                   0    0.042905                       0.0
28819                   3    0.039698                       8.0
28820                   0    0.018346                       2.0
28821                   1    0.924899                       0.0
28822                   0    0.035456                       0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                        …                       …                …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic
0          0
1          0
2          0
3          0
4          0
…      …
28818      0
28819      1
28820      1
28821      0
28822      0
```

```
[28823 rows x 18 columns]
```

[3]:
```python
## Clean data
df = df.dropna(axis=0) # removes rows containing missing values
df = df.drop_duplicates() # removes duplicate rows leaving just onedf.shape
```

[4]:
```python
## Step 2: Create new variable for total toxic replies in a conversation
df['Total_toxic_replies'] = df['Num_toxic_direct_replies'] +␣
 ↪df['Num_toxic_nested_replies']
## Create another binary column Toxic_conversation where if the total toxic␣
 ↪replies is at least 1 Toxic_conversation will be 1, otherwise it is 0
df['Toxic_conversation'] = (df['Total_toxic_replies'] > 0).astype(int)
df
```

[4]:
```
       Unnamed: 0                Tweet  Followers  Friends  Num_tweets  \
0               0  1441497018807906305       1198    605.0        6166
1               1  1431812786099613699        608   1259.0        1811
2               2  1426644707313135617        173    167.0        4306
3               3  1431814908425998337       2540    222.0        6008
4               4  1432862687533441027       4439     11.0        9985
…             …                    …        …      …          …
28818       28815  1431029315274162181      16182   1150.0       58245
28819       28816  1437970083087851520     688077    636.0       12308
28820       28817  1428110541093052418      57068   1225.0       25131
28821       28818  1430722665514377219         66    121.0         549
28822       28819  1441165693974503442        169    276.0        2910

       Verified  Listed_count Location  Age  Length  Num_users  \
0         False             1     True    7       6          4
1         False             5     True    2       2          2
2         False             0     True    0       4          2
3         False             0     True    9       2          2
4         False            55     True    2       4          3
…           …             …      …    …      …          …
28818      True           685    False   12       2          2
28819      True           361    False   10     247        218
28820     False             0     True    5      22         21
28821     False             3     True    1       4          3
28822     False             7     True    0       2          2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2    0.235235                       0.0
1                       0    0.085582                       0.0
2                       2    0.076877                       0.0
3                       0    0.095684                       0.0
4                       1    0.165919                       0.0
…                      …         …                        …
```

```
28818                    0    0.042905                    0.0
28819                    3    0.039698                    8.0
28820                    0    0.018346                    2.0
28821                    1    0.924899                    0.0
28822                    0    0.035456                    0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                         …                        …                 …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic  Total_toxic_replies  Toxic_conversation
0          0                  0.0                   0
1          0                  0.0                   0
2          0                  0.0                   0
3          0                  0.0                   0
4          0                  0.0                   0
…      …                    …                 …
28818      0                  0.0                   0
28819      1                 10.0                   1
28820      1                  2.0                   1
28821      0                  0.0                   0
28822      0                  0.0                   0

[28818 rows x 20 columns]
```

[5]:
```python
## Split Data into training (80%) and testing (20%), using Length, Num_users,␣
↪Toxicity, Num_author_replies, Verified, and Age as features
X_Toxic =␣
↪df[['Length','Num_users','TOXICITY_x','Num_author_replies','Verified',␣
↪'Age']]
y_Toxic = df['Toxic_conversation']
X_train_Toxic, X_test_Toxic, y_train_Toxic, y_test_Toxic =␣
↪train_test_split(X_Toxic, y_Toxic, test_size=0.2, random_state=42)
```

[6]:
```python
## Step 3: Train and Evaluate Models with 5-Fold Cross Validation
# Initialize Logistic Regression
logreg = LogisticRegression(max_iter=200)
```

```
# Perform cross-validation and get F1 scores
f1_scores_logreg = cross_val_score(logreg, X_train_Toxic, y_train_Toxic, cv=5,␣
 ↪scoring='f1_weighted')

# Print F1 scores across folds and mean F1 score
print("Logistic Regression F1 scores across folds:", f1_scores_logreg)
print("Logistic Regression mean F1 Score:", f1_scores_logreg.mean())

# Perform cross-validation and get Accuracy scores
accuracy_scores_logreg = cross_val_score(logreg, X_train_Toxic, y_train_Toxic,␣
 ↪cv=5, scoring='accuracy')

# Print accuracy scores across folds and mean accuracy score
print("Logistic Regression accuracy scores across folds:",␣
 ↪accuracy_scores_logreg)
print("Logistic Regression mean accuracy score:", accuracy_scores_logreg.mean())
```

```
Logistic Regression F1 scores across folds: [0.71965413 0.73501625 0.72608172
0.72850329 0.72459843]
Logistic Regression mean F1 Score: 0.7267707626847467
Logistic Regression accuracy scores across folds: [0.77488614 0.78052483
0.78139232 0.78139232 0.77744035]
Logistic Regression mean accuracy score: 0.7791271932486259
```

[7]:
```
## Initialize Support Vector Machine
svm = SVC()

# Perform cross-validation and get F1 scores
f1_scores_svm = cross_val_score(svm, X_train_Toxic, y_train_Toxic, cv=5,␣
 ↪scoring='f1_weighted')

# Print F1 scores across folds and mean F1 score
print("Support Vector Machine F1 scores across folds:", f1_scores_svm)
print("Support Vector Machine mean F1 Score:", f1_scores_svm.mean())

# Perform cross-validation and get accuracy scores
accuracy_scores_svm = cross_val_score(svm, X_train_Toxic, y_train_Toxic, cv=5,␣
 ↪scoring='accuracy')

# Print accuracy scores across folds and mean accuracy score
print("Support Vector Machine accuracy scores across folds:",␣
 ↪accuracy_scores_svm)
print("Support Vector Machine mean accuracy score:", accuracy_scores_svm.mean())
```

```
Support Vector Machine F1 scores across folds: [0.71689119 0.72315064 0.7258758
0.73033405 0.71827069]
Support Vector Machine mean F1 Score: 0.7229044760578092
```

```
Support Vector Machine accuracy scores across folds: [0.77358491 0.77531989
0.77813923 0.78269356 0.77331887]
Support Vector Machine mean accuracy score: 0.7766112912111047
```

[9]:
```
# Initialize Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Perform cross-validation and get F1 scores
f1_scores_rf = cross_val_score(rf, X_train_Toxic, y_train_Toxic, cv=5,␣
↪scoring='f1_weighted')

# Print F1 scores across folds and mean F1 score
print("Random Forest - F1 Scores across folds:", f1_scores_rf)
print("Random Forest - Mean F1 Score:", f1_scores_rf.mean())
```

```
Random Forest - F1 Scores across folds: [0.71878245 0.72432428 0.73564806
0.72778668 0.72893291]
Random Forest - Mean F1 Score: 0.727094877620002
```

[10]:
```
## Step 4: Retrain the Best Model on Full Training Data and test it using the␣
↪testing dataset you set aside in step #2. Print out the accuracy and F1.
logreg.fit(X_train_Toxic, y_train_Toxic) # train model on the whole training␣
↪dataset

y_pred_Toxic = logreg.predict(X_test_Toxic)

test_accuracy_Toxic = accuracy_score(y_test_Toxic, y_pred_Toxic)
test_f1_Toxic = f1_score(y_test_Toxic, y_pred_Toxic, average='weighted')  #␣
↪Weighted for multi-class

print("Test Accuracy of logistic regression model:" + str(test_accuracy_Toxic))
print("Test F1 Score of logistic regression model:"  + str(test_f1_Toxic))
```

```
Test Accuracy of logistic regression model:0.7746356696738376
Test F1 Score of logistic regression model:0.7222816417658159
```

0.2 Part b: Regularization

[11]:
```
## Step 1: Split original data frame into training (80%) and testing (20%)␣
↪again, using more features, (Length, Num_users, Toxicity,␣
↪Num_author_replies, Verified, Age, Followers, Friends, Num_tweets, Location,␣
↪Listed_count(
X_Toxic =␣
↪df[['Length','Num_users','TOXICITY_x','Num_author_replies','Verified',␣
↪'Age', 'Followers', 'Friends', 'Num_tweets', 'Location', 'Listed_count']]
y_Toxic = df['Toxic_conversation']
```

```
X_train_Toxic, X_test_Toxic, y_train_Toxic, y_test_Toxic =␣
↪train_test_split(X_Toxic, y_Toxic, test_size=0.2, random_state=42)
```

[12]:
```
## Step 2: Fit a Logistic Regression model and evaluate its performance using␣
↪5-fold cross- validation
# Initialize Logistic Regression
logreg = LogisticRegression(max_iter=200)

# Perform cross-validation and get f1 scores
f1_scores_logreg = cross_val_score(logreg, X_train_Toxic, y_train_Toxic, cv=5,␣
↪scoring='f1_weighted')

# Print F1 scores across folds and mean F1 score
print("Logistic Regression F1 scores across folds:", f1_scores_logreg)
print("Logistic Regression mean F1 Score:", f1_scores_logreg.mean())

# Perform cross-validation and get Accuracy scores
accuracy_scores_logreg = cross_val_score(logreg, X_train_Toxic, y_train_Toxic,␣
↪cv=5, scoring='accuracy')

# Print accuracy scores across folds and mean accuracy score
print("Logistic Regression accuracy scores across folds:",␣
↪accuracy_scores_logreg)
print("Logistic Regression mean accuracy score:", accuracy_scores_logreg.mean())
```

```
Logistic Regression F1 scores across folds: [0.6575518  0.65436314 0.66034031
0.7098215  0.66259574]
Logistic Regression mean F1 Score: 0.6689344952301355
Logistic Regression accuracy scores across folds: [0.74647582 0.74799393
0.74929516 0.74517458 0.65292842]
Logistic Regression mean accuracy score: 0.7283735818007584
```

[13]:
```
## Step 2: apply L1 regularization (Lasso), L2 regularization (Ridge), and␣
↪Elastic Net regularization to the Logistic Regression model. Perform 5-fold␣
↪cross-validation to compare the performance of each regularization method
# Logistic Regression with Lasso method
from sklearn.metrics import classification_report
logreg_lasso = LogisticRegression(penalty='l1', solver='liblinear',␣
↪multi_class='ovr')

# Use cross validation to get f1 and accuracy scores
f1_logreg_lasso = cross_val_score(logreg_lasso, X_train_Toxic, y_train_Toxic,␣
↪cv=5, scoring='f1_weighted')
accuracy_logreg_lasso = cross_val_score(logreg_lasso, X_train_Toxic,␣
↪y_train_Toxic, cv=5, scoring='accuracy')
```

```
print("Lasso Regularization Avg Accuracy:", accuracy_logreg_lasso.mean())
print("Lasso Regularization Avg F1 Score:", f1_logreg_lasso.mean())

asso_coeffs = lasso_coeffs.sort_values(by='Coefficient', key=abs,␣
 ↪ascending=False)
print(
```

```
Lasso Regularization Avg Accuracy: 0.7793874687098803
Lasso Regularization Avg F1 Score: 0.7276225496944569
```

[20]:
```
## Logistic Regression with Ridge method
logreg_ridge = LogisticRegression(penalty='l2', C=0.1, solver='saga',␣
 ↪multi_class='ovr')

# Use cross validation to get accuracy and F1 scores
accuracy_ridge = cross_val_score(logreg_ridge, X_train_Toxic, y_train_Toxic,␣
 ↪cv=5, scoring='accuracy')
f1_ridge = cross_val_score(logreg_ridge, X_train_Toxic, y_train_Toxic, cv=5,␣
 ↪scoring='f1_weighted')

print("Lasso Regularization Avg Accuracy:", accuracy_ridge.mean())
print("Lasso Regularization Avg F1 Score:", f1_ridge.mean())
```

```
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
```

```
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
```

```
Lasso Regularization Avg Accuracy: 0.751539838479238
Lasso Regularization Avg F1 Score: 0.6592913803100651
```

```
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
```

[21]:
```
# Logistic Regression with Elastic Net regularization (Combination of L1 and L2)
logreg_elastic = LogisticRegression(penalty='elasticnet', solver='saga',␣
 ↪l1_ratio=0.5, multi_class='ovr')

# Use cross validation to get accuracy and F1 scores
accuracy_elastic = cross_val_score(logreg_elastic, X_train_Toxic,␣
 ↪y_train_Toxic, cv=5, scoring='accuracy')
f1_elastic = cross_val_score(logreg_elastic, X_train_Toxic, y_train_Toxic,␣
 ↪cv=5, scoring='f1_weighted')

print("Elastic Regularization Avg Accuracy score:", accuracy_elastic.mean())
print("Elastic Regularization Avg F1 Score:" , f1_elastic.mean())
```

```
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
```

```
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
```

```
Elastic Regularization Avg Accuracy score: 0.751539838479238
Elastic Regularization Ave F1 Score 0.6593156766419529
```

```
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
```

0.3 Part 3: Comparison of the results

The models all had similar cross valuation scores for f1 and accuracy, but lasso had the highest of the three techniques at . Lasso can bring coefficients down to 0 for features that are not useful in predicting results. It likely performed the best because there were some factors like location that could be completely left out. It also likely outperformed the elastic regularization because there were a number of different features that were not relevant. All of the regularization techniques did increase the performance from the original model, but ridge and elastic did have similar scores to the original model. I think these results show us that adding all of these features isn't neccesarily going to help the model perform better, because some of the features were not necessary and that was shown by the performance of the model after using lasso regularization.

[ ]:

```

```