# Machine Learning Final Project Spring 2023

*Kaihil Patel, Dhruv Pandya, Courtney Nguyen, Aaron Parson*

```
#Machine Learning Final Project Spring 2023
#Kaihil Patel, Dhruv Pandya, Courtney Nguyen, Aaron Parson
```

Double-click (or enter) to edit

# Section 1: Data Preprocessing

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import io
from google.colab import files

uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving political_social_media.csv to political_social_media.csv
```

```
# Load the data and extract the bias and text columns
df = pd.read_csv('political_social_media.csv', encoding='latin1', usecols=['bias', 'text'])
```

# Section 2: Data Splits

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

# Convert the text into numeric vectors using CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['text'])
```

```
# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, df['bias'], test_size=0.2, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_train, y_train, test_size=0.5, random_state=42)
```

# Section 3: Build Classifiers

```
#Build the Classifier

#Pick 2 of (Perceptron, *Logistic regression*, *SVM*, Feed-forward neural networks)
#Report Prediction accuracies using default parameters
```

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

log = LogisticRegression()
log.fit(X_train, y_train)

trn_acc = log.score(X_train, y_train)
val_acc = log.score(X_test, y_test)

print("Training accuracy: {}".format(trn_acc))
print("Validation accuracy: {}".format(val_acc))
```

```
Training accuracy: 0.98925
Validation accuracy: 0.992
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
# SVM
from sklearn.svm import SVC

clf = SVC()
clf.fit(X_train, y_train)

y_val_pred = clf.predict(X_val)
y_test_pred = clf.predict(X_test)

val_accuracy = sum(y_val_pred == y_val) / len(y_val)
test_accuracy = sum(y_test_pred == y_test) / len(y_test)

print("Validation accuracy:", val_accuracy)
print("Test accuracy:", test_accuracy)
```

```
Validation accuracy: 0.8265
Test accuracy: 0.815
```

## Section 4: Hyper-parameter tuning

```
#Classifier 1: SVM
    #Pick 3 hyper parameters, report their pre defined values
    #report the prediction accuracy of the validation set on each combination of the hyper-parameters
    #report the prediction accuracy of the test set with the best combination of the hyper-parameters
```

```
# SVM Kernel = Sigmoid

clf = SVC(kernel='sigmoid')
clf.fit(X_train, y_train)

y_val_pred = clf.predict(X_val)
y_test_pred = clf.predict(X_test)

val_accuracy = sum(y_val_pred == y_val) / len(y_val)
test_accuracy = sum(y_test_pred == y_test) / len(y_test)

print("Validation accuracy Sigmoid:", val_accuracy)
print("Test accuracy Sigmoid:", test_accuracy)
```

```
Validation accuracy Sigmoid: 0.741
Test accuracy Sigmoid: 0.7225
```

The default kernel for SVM algorithm is the RBF kernel. Here we changed it to the sigmoid.

```
# SVM gamma = 2

clf = SVC(gamma=2)
clf.fit(X_train, y_train)

y_val_pred = clf.predict(X_val)
y_test_pred = clf.predict(X_test)

val_accuracy = sum(y_val_pred == y_val) / len(y_val)
test_accuracy = sum(y_test_pred == y_test) / len(y_test)

print("Validation accuracy Gamma:", val_accuracy)
print("Test accuracy Gamma:", test_accuracy)
```

```
Validation accuracy Gamma: 1.0
Test accuracy Gamma: 1.0
```

The default value for gamma is scale. If this is passed, then it uses 1 as the value of gamma. Here we changed it to 2.

```
# SVM Degree = 3

clf = SVC(degree=3)
clf.fit(X_train, y_train)

y_val_pred = clf.predict(X_val)
y_test_pred = clf.predict(X_test)

val_accuracy = sum(y_val_pred == y_val) / len(y_val)
test_accuracy = sum(y_test_pred == y_test) / len(y_test)

print("Validation accuracy Degree:", val_accuracy)
print("Test accuracy Degree:", test_accuracy)
```

```
Validation accuracy Degree: 0.8265
Test accuracy Degree: 0.815
```

```
#Classifier 2: Linear Regression
    #Pick 3 hyper parameters, report their pre defined values
    #report the prediction accuracy of the validation set on each combination of the hyper-parameters
    #report the prediction accuracy of the test set with the best combination of the hyper-parameters
```

```
#Logistic Regression Change Solver

log = LogisticRegression(solver='saga')
log.fit(X_train, y_train)
trn_acc = log.score(X_train, y_train)
val_acc = log.score(X_test, y_test)
print("Training accuracy Solver: {}".format(trn_acc))
print("Validation accuracy Solver: {}".format(val_acc))
```

```
Training accuracy Solver: 0.74775
Validation accuracy Solver: 0.745
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was reached which means the coef_ d
  warnings.warn(
```

The default value for solver is lbfgs, here we changed it to saga.

```
#Logistic Regression Change Fit Intercept

log = LogisticRegression(fit_intercept=False)
log.fit(X_train, y_train)
trn_acc = log.score(X_train, y_train)
val_acc = log.score(X_test, y_test)
print("Training accuracy Fit Intercept: {}".format(trn_acc))
print("Validation accuracy Fit Intercept: {}".format(val_acc))
```

```
Training accuracy Fit Intercept: 0.9915
Validation accuracy Fit Intercept: 0.993
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

The default value for fit_intercept is True, here we changed it to False.

```
#Logistic Regression Change C

log = LogisticRegression(C=0.1)
log.fit(X_train, y_train)
trn_acc = log.score(X_train, y_train)
val_acc = log.score(X_test, y_test)
print("Training accuracy C: {}".format(trn_acc))
print("Validation accuracy C: {}".format(val_acc))
```

```
Training accuracy C: 0.85925
Validation accuracy C: 0.8595
```

The default value for C is 1.0, here we changed it to 0.1.

## Section 5: Analysis

```
# Analysis

# * Based on the results above, which classifier is better, and why?
# * For further improvement on classification accuracy,
#   what strategies that you can use and why do you think they will be helpful?
#
#   * List at least two strategies for analysis
#   * You don’t have to implement these strategies.
#     Therefore, your justification should not be “we tried them and they worked”
#   * If your strategy involves using large language models,
#     you should show a good understanding about how to use them,
#     in addition the justification about why they could help.
```

The best classifier is the SVM Classifier with the hyperparameter gamma = 2. This is because it determines the shape of the decision boundary. A lower gamma generally causes the decision boundary to be smoother, while a higher decision boundray to be more complex. However, higher values of gamma can lead to overfitting. The validation accuracy and testing accuracy were 1 for the SVM(gamma=2). This denotes that the problem does not require a complex boundary to make decisions. Although we have a perfect validation accuracy, some other strategies that can improve an accuracy of the model is changing the data such as adding more noise. This can make the dataset larger and therefore giving a better prediction. Another way to increase the accuracy is further feature engineering. We can change the degrees and the gamma function which could increase the accuracy further.